National Library of Canada
Bibliothèque nationale du Canada

Canadian Theses Service
Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, tests publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30.

NL-339 (r. 88/04)

Canada

THE UNIVERSITY OF ALBERTA

MULTI-STEP ADAPTIVE PREDICTIVE CONTROL
WITH
DISTURBANCE MODELING

by

©NAGESWARA RAO SRIPADA

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH
IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE
OF DOCTOR OF PHILOSOPHY
IN
PROCESS CONTROL

DEPARTMENT OF CHEMICAL ENGINEERING

EDMONTON, ALBERTA
SPRING 1988

Permission has been granted to the National Library of Canada to microfilm this thesis and to lend or sell copies of the film.

The author (copyright owner) has reserved other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without his/her written permission.

L'autorisation a été accordée à la Bibliothèque nationale du Canada de microfilmer cette thèse et de prêter ou de vendre des exemplaires du film.

L'auteur (titulaire du droit d'auteur) se réserve les autres droits de publication; ni la thèse ni de longs extraits de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation écrite.

ISBN 0-315-42753-1

THE UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR NAGESWARA RAO SRIPADA

TITLE OF THESIS MULTI-STEP ADAPTIVE PREDICTIVE CONTROL WITH DISTURBANCE MODELING

DEGREE FOR WHICH THESIS WAS PRESENTED DOCTOR OF PHILOSOPHY

YEAR THIS DEGREE GRANTED SPRING 1988

Permission is hereby granted to THE UNIVERSITY OF ALBERTA LIBRARY to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without the author's written permission.

(SIGNED) NRSripada

PERMANENT ADDRESS:
C/o Dr. S. V. S. Murty
PRL, Navrangpura
Ahmedabad, India 380 009

DATED April 25th 1988.

THE UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research, for acceptance, a thesis entitled MULTI-STEP ADAPTIVE PREDICTIVE CONTROL WITH DISTURBANCE MODELING submitted by NAGESWARA RAO SRIPADA in partial fulfilment of the requirements for the degree of DOCTOR OF PHILOSOPHY in PROCESS CONTROL.

D.G. Fisher

Supervisor

V. G. Gourishankar

P.J. Crickmore

D.W. Clarke

External Examiner

Date April 25, 1988

To my nagu and to my parents

Abstract

The objective of this work was to apply modern control theory and numerical methods to practical applications characterized by difficulties such as non-square MIMO processes, time-delays, constraints, non-stationary disturbances and unknown mathematical models. The results fall into several distinct areas: a predictive control system (MOCCA) for constrained MIMO processes characterized by step responses; expert systems for real-time, sensor-based applications; an optimal factorization of transfer function matrices that contain arbitrary time-delays; an improved least squares (ILS) algorithm for parameter estimation; an adaptive predictive control system that includes explicit adaptive disturbance modeling plus a multi-step cost function. Although the first three results can be used independently, they are all incorporated into the overall design of the adaptive controller which is the main result of this thesis.

The Multivariable, Optimal, Constrained Control Algorithm (MOCCA) was inspired by the very successful Dynamic Matrix Control (DMC) algorithm developed and applied by Shell Oil Ltd.

Two expert systems were developed and evaluated experimentally to demonstrate that expert systems could be successfully used in process applications. The first learned key parameters on-line and approximated time-optimal servo control. The second illustrated feedback control in the presence of constraints.

The interactor matrix characterizes the delay structure in MIMO discrete systems and leads to an optimal

factorization of MIMO transfer function matrices containing arbitrary time-delays. The resulting interactor factorization provides a basis for designing MIMO time-delay compensators, predictive and feedforward controllers.

The basic least squares algorithm was extended to include filtering, normalization and scaling of the zero-mean regressor data. A variable forgetting factor was used to maintain the trace of the covariance matrix constant and an on/off feature was included to turn identification off during periods of insufficient excitation. Numerous examples illustrate potential problems and the effectiveness of the improved (ILS) algorithm.

The explicit, Multi-step Adaptive Predictive Controller (MAPC) can be compared to the Generalized Predictive Controller (GPC) and contains explicit features to deal with noise, load-disturbances and unmodelled dynamics. MAPC uses a separate residual model in estimation and controller synthesis, a Kalman filter for prediction, and an expert system to control identification and supervise overall operation.

Acknowledgement

The author wishes to thank his thesis supervisor, Dr. D. Grant Fisher, for his constant encouragement and help, and for his consideration throughout the author's graduate work.

Table of Contents

Chapter Page

1. Introduction 1

References 7

2. An AI application for process regulation and servo control 8

2.1 Introduction 8

2.1.1 Description of the experimental equipment 9

2.1.2 Data acquisition 11

2.1.3 Knowledge acquisition 11

2.1.4 Experimental procedure 11

2.2 Servo control 12

2.2.1 Knowledge base 14

2.2.2 Discussion of results 17

2.3 Regulatory control 22

2.3.1 Knowledge base 29

2.3.2 Discussion of results 33

2.4 Distinguishing characteristics of RTSB applications of AI 43

2.5 Practical aspects of feedback control 46

2.6 Conclusions 48

References 49

3. Multivariable, Optimal, Constrained Control Algorithm (MOCCA) 50

3.1 Introduction 50

3.2 A functional overview of MOCCA 52

3.3 MOCCA: Forward path for servo control 55

3.3.1 Model formulation 55

3.3.1.1 Model formulation for SISO processes 55

3.3.1.2 Model formulation for MIMO processes 61

3.3.1.3 Model formulation for time-delay systems 62

3.3.1.4 Model formulation for measured disturbances 63

3.3.2 MOCCA control calculation for servo control 64

3.3.2.1 MOCCA with constraints 68

3.4 MOCCA: Feedback path for regulatory control 69

3.4.1 Feedback interpretation of regulatory control 70

3.4.2 Feedforward interpretation of regulatory control 70

3.4.3 Formulation of the feedback predictor 71

3.4.4 Feedback filter formulation 73

3.5 MOCCA Supervisory system 73

3.6 MOCCA, DMC and MAC: Literature Survey 74

3.6.1 Dynamic Matrix Control (DMC) 75

3.6.2 Model Algorithmic Control (MAC) 76

3.6.3 Other approaches 77

3.6.4 Industrial applications 77

3.7 MOCCA: Evaporator application 78

3.7.1 MOCCA parameters 80

3.7.2 Evaporator run #1 81

3.7.3 Evaporator run #2 84

3.7.4 Evaporator run #3 85

3.7.5 Evaporator run #4 85

3.8 Conclusions 86

References 87

4. Control of discrete MIMO systems with time-delays using the interactor matrix 89

4.1 Introduction 89

4.2 Properties of time-delay systems 92

4.2.1 SISO systems 93

4.2.2 MIMO systems 95

4.3 Time delay compensation in feedback systems 99

4.3.1 General formulation 100

4.3.2 SISO Smith predictor 102

4.3.3 MIMO time-delay compensators 103

4.3.4 Interactor factorization 105

4.3.4.1 Decoupled response when the inverse interactor is triangular 110

4.3.4.2 Comparison with T.T. factorization of IMC 112

4.3.5 A binary distillation column example 116

4.3.6 A recycle reactor example 125

4.4 Feedforward control 130

4.5 Model following 137

4.6 MIMO predictive control 139

4.7 Conclusions 141

References 144

Appendix 4.1 146

5. Improved least squares identification 151

5.1 Introduction 151

5.1.1 Turn-off 152

5.1.2 Covariance blowup 152

5.1.3 Parameter drift and bursting 153

5.1.4 Excitation 153
5.1.5 Biased estimates 154
5.1.6 Theoretical properties 154
5.1.7 Numerical properties 155
5.2 Improved least squares estimation 155
5.2.1 Process representation 156
5.2.2 Basic least squares algorithm 158
5.2.3 Formulation of the improved least squares algorithm 159
5.3 Convergence behaviour of the improved least squares estimator 170
5.4 Literature review 175
5.4.1 Non-zero mean data 175
5.4.2 Model Process Mismatch 177
5.4.3 Tracking time varying parameters 178
5.5 Conclusions 180
References 183
Appendix 5.1 186
Appendix 5.2 193
6. Multi-step Adaptive Predictive Controller (MAPC) with disturbance modeling and Kalman filter prediction 243
6.1 Introduction 243
6.1.1 A review of models used in self-tuning control 247
6.1.2 An overview of proposed MAPC 254
6.2 Process representation 259
6.2.1 One-step-ahead Kalman filter predictor 268
6.3 Model identification and parameter estimation ...271

6.3.1 Estimation of the process model parameters 271

6.3.2 Estimation of the residual model parameters 273

6.4 Output prediction 275

6.5 The multi-step predictive controller 283

6.5.1 Choice of the controller parameters 288

6.5.2 Special cases of the proposed controller 291

6.6 The supervisory system 298

6.7 The AI executive 301

6.7.1 Some example rules used in the supervision of the model identification step of MAPC 304

6.8 Convergence properties of MAPC 309

6.9 Literature review 312

6.10 Simulated, single-input, single-output, applications 315

6.11 Conclusions 319

References 322

Appendix 6.1 326

7. Conclusions and recommendations for future work 351

7.1 Conclusions 351

7.2 Recommendations for future work 355

8. Nomenclature 359

8.1 Technical abbreviations 359

8.2 Nomenclature for Chapter Two 360

8.3 Nomenclature for Chapter Three 361

8.4 Nomenclature for Chapter Four 364

8.5 Nomenclature for Chapter Five 366

8.6 Nomenclature for Chapter Six 368

List of Tables

Table Page

2.1 Summary of the procedure for feedback control using AI plus fuzzy logic approach........................32

3.1 MOCCA Features...52

4.1 A list of models used in the illustrative examples with interactor factorization.......................115

4.2 The predictors used in OR and IF Smith predictors for the distillation column and reactor examples....117

4.3 PI controller parameters for the distillation column example.......................................120

4.4 PI controller parameters for the reactor example....128

5.1 An overview of the simulated example runs...........194

5.2 A list of process models used in the simulations....195

5.3 A list of controllers used in the simulations.......196

6.1 MAPC Features.......................................257

List of Figures

Figure Page

2.1 Schematic diagram of the equipment used for experimental evaluation of AI based controllers......10

2.2 Time-optimal, open-loop state driving of the controlled variable, y(k), from one setpoint to another showing the effect of errors in e^* 13

2.3 State transition diagram representation of the procedure for optimal servo control.................16

2.4 A s~~et of experim~~ental runs illustrating how e^* is learned on-line and the corresponding improvement in performance......................................18

2.5 A set of experimental runs showing how N and e^* are learned on-line......................................19

2.6 Experimental response of the process to a setpoint change using a conventional, fixed-gain PI controller21

2.7 Qualitative representation of the three different operating regions for the AI based regulatory controller..23

2.8 Block diagram of a standard feedback control system..25

2.9 Fuzzy set membership functions for the magnitude of the error $e:=y_{sp}-y$ 27

2.10 Fuzzy set membership functions for the magnitude of the slope of the response.............................28

2.11 Fuzzy set membership functions defining the magnitude of the changes in the manipulated variable.29

2.12 Experimental process response near the setpoint using AI control......................................34

2.13 Experimental process response near the setpoint using high-gain PI controller..........................35

2.14 Experimental process response for a disturbance using AI control......................................37

2.15 Experimental process response for a disturbance using AI control, when a constraint is active........38

2.16 Experimental process response for a disturbance using a medium gain PI controller....................39

2.17 Experimental process response for a setpoint change using AI control..40

2.18 Experimental process response for a setpoint change using AI control, when the new setpoint is near a constraint..42

2.19 Experimental process response for a setpoint change using high-gain PI controller........................43

3.1 Schematic representation of MOCCA showing process, forward path, feedback path and supervisory system...53

3.2 Discrete, non-functional representation of the process response, y, to a step change in input; i.e. y(t) is approximated by $\{a_1, a_2, \ldots, a_N\}$........56

3.3 Schematic diagram of the double effect evaporator....79

3.4 Simulated responses of the evaporator to a 20% increase in the C_2 setpoint 82

3.5 Simulated responses of the evaporator to a 20% increase in feed flow rate..........................84

4.1 Open-loop inverse model control for perfect setpoint following.................................95

4.2 Structure of the generalized Smith predictor........101

4.3 Internal model controller (IMC)...................108

4.4 Use of pre- and/or post-compensators for achieving decoupling...111

4.5 Simulated responses of the distillation column for dead-beat control and for setpoint changes..........119

4.6 Simulated responses of the distillation column for delay compensated multi-loop PI control and for setpoint changes.................................121

4.7 Output of the predictor in the distillation column example for multi-loop PI control and for setpoint changes...122

4.8 Simulated responses of the distillation column for delay compensated, multi-loop PI control and for disturbance changes...............................123

4.9 Output of the predictor in the distillation column example under multi-loop PI control for disturbance changes....124

4.10 Schematic diagram of the two-stage chemical reactor.126

4.11 Simulated responses of the chemical reactor for delay compensated multi-loop PI control and for a disturbance change....129

4.12 Feedforward controller....131

4.13 Simulated responses in the feedforward control example for a measured disturbance of $v=(1,0)^T$......135

4.14 Simulated examples in the feedforward control example for a measured disturbance of $v=(0,1)^T$......136

4.15 A model following control system....138

5.1 Effect of turn-off on the parameter estimates.......198

5.2 Trace of P when turn-off occurs....199

5.3 System output....201

5.4 Covariance windup with a constant λ....202

5.5 System output showing bursts....204

5.6 Parameter estimates obtained with covariance windup.205

5.7 Covariance windup with a constant λ....206

5.8 System output showing bursts....208

5.9 Parameter estimates obtained with covariance windup.209

5.10 Covariance windup with variable forgetting factor of Fortescue....210

5.11 Variable forgetting factor of Fortescue....211

5.12 Effect of d.c. bias on estimated parameters.........213

5.13 Effect of an ill-conditioned P on the estimated parameters in RLS....215

5.14 Effect of an ill-conditioned P on the estimated parameters in ILS without scaling....217

5.15 Effect of model-process mismatch on the estimated parameters in RLS....219

5.16 System output....................................222

5.17 Parameter drifting due to lack of excitation........223

5.18 Estimated parameters when mean-deviational data are used in RLS..225

5.19 Parameter estimates obtained using RLS with data scaling...227

5.20 Parameter estimates obtained using ILS without the on/off criteria to show the effect of scaling.......229

5.21 Effect of low-pass filtering on the estimated parameters in RLS...................................231

5.22 System output....................................233

5.23 Effect of using the on/off criteria in ILS on the estimated parameters.................................234

5.24 Trace of P before (dashed line) and after (continuous line) scaling in ILS with on/off........235

5.25 Parameter estimates obtained with tr P=1 and without scaling......................................238

5.26 Parameter estimates obtained with tr P=1 and with scaling...239

5.27 Parameter estimates obtained with tr P=4 and without scaling......................................240

5.28 Parameter estimates obtained with tr P=4 and with scaling...241

5.29 Parameter estimates obtained using Goodwin"s constant trace algorithm with tr P=1...............242

6.1 A high-level, schematic block diagram of MAPC.......258

6.2 A high-level block diagram of non-adaptive MAPC.....280

6.3 A detailed block diagram of non-adaptive MAPC.......281

6.4 A high-level block diagram of MAPC.................282

6.5 A detailed block diagram of MAPC...................283

6.6 Smith predictor....................................294

6.7 Kalman filter predictor based feedback controller...295

6.8 Smith predictor with a filter......................296

6.9 Internal model controller (IMC)....................298

6.10 Performance of the basic, non-adaptive MAPC in the presence of noise....................................331

6.11 Performance of the basic, non-adaptive GPC in the presence of noise....................................332

6.12 Performance of the basic, non-adaptive MAPC in the presence of load-disturbances (an exact internal model is used)....................................335

6.13 Performance of the basic, non-adaptive MAPC in the presence of load-disturbances (the disturabance model is an integrator)...........................336

6.14 Performance of the basic, non-adaptive GPC in the presence of load-disturbances.......................337

6.15 Performance of MAPC in the presence of noise........340

6.16 Parameter estimates in MAPC (the parameter values are normalized).....................................341

6.17 Performance of GPC in the presence of noise.........342

6.18 Parameter estimates in GPC (the parameter values are normalized).....................................343

6.19 Performance of MAPC in the presence of load-disturbances (case (i), N_y=8)..................346

6.20 Performance of MAPC in the presence of load-disturbances (case (ii), N_y=2).................347

6.21 Performance of MAPC in the presence of unmodelled dynamics.......................................350

1. Introduction

The original objectives of this thesis were goal-oriented and two pronged:

1) to identify and solve some of the practical control problems encountered in industrial processes;
2) to determine how some of the latest control and computer techniques could be applied to practical industrial processes.

Some of the industrial control problems that did not appear to be adequately handled by conventional computer control systems (e.g. Honeywell's TDC.2000) were:

1) MIMO processes, i.e. interactions and decoupling;
2) time-delays;
3) constraints on manipulated and controlled variables;
4) development and utilization of process information;
5) heuristic guidelines and rules used by humans.

There have been several significant developments over the past few years in the area of control theory and computer techniques. For example, some of the tools, techniques and theory that are potentially relevant to problems such as those listed above are:

1) predictive control algorithms such as Shell's DMC;
2) constrained optimization techniques;
3) improved process identification and adaptive control schemes;
4) artificial intelligence and expert systems.

The scope and direction of the thesis has changed as it progressed but can now be divided into three sections. The first shows how expert systems can be used to include heuristic knowledge and operational experience in the control strategies for industrial processes. The second section shows how non-functional step response information can be incorporated into a practical Multivariable Optimal Constrained Control Algorithm (MOCCA) and develops an optimal factorization for handling MIMO systems with time-delays. The last section deals with parameter estimation and adaptive control.

Expert systems (ES): Expert systems are receiving a lot of attention because they provide a convenient framework to incorporate heuristic logic, operator experience etc. into control systems. The field of Artificial Intelligence (AI) is rather large and a lot of literature has already accumulated. A selected list of annotated references on AI and Knowledge Based Systems (KBS) was prepared and is available as an internal report (Sripada (1986)) and may be useful as a starting point. A brief introduction to AI and KBS was presented in Sripada *et al.* (1985) along with an outline of possible applications of these systems in solving problems in the process industry. The real-time and sensor-based nature of these applications was emphasized. In particular two applications in real-time process control were discussed and illustrated with experimental results.

Chapter 2 outlines a specific real-time application of expert systems for servo and regulatory control of SISO processes. The servo controller is based on the time-optimal bang-bang control strategy and the objective of the expert system is to learn the switching parameters on-line from observed response behaviour. The regulatory controller is based on "Situation => Action" type heuristic logic. An important feature of the regulatory controller is that it varies its control action in the sense that it provides smooth control action near the setpoint (cf. a noise dead-zone); normal control action for typical disturbances; and vigorous control action to ensure that a constraint on the output is not violated. It is hard to achieve such control behaviour using simple PID controllers and hence the expert systems approach may have some practical applications although the work is intended more as a demonstration of what can be accomplished using AI. The AI programming language PROLOG was used in implementation of these controllers. Some important characteristics of real-time sensor based AI applications are also summarized in this chapter.

MOCCA and time-delay factorization: When process information is available it is relatively straightforward to design an appropriate control system. A number of techniques have been developed in the literature such as frequency domain design techniques based on transfer function models and optimal control design methods based on state-space models. However the success of these methods depends on minimal (or parametric) models which are often hard to develop. The design procedure becomes especially complicated when there are multiple time-delays in the system. Therefore attention has been given to developing control systems that are based on models that can be readily obtained from plant data and on handling time-delays.

The MOCCA scheme presented in Chapter 3 arose from the need to develop easy to construct, robust and practical non-adaptive control schemes that can be applied to multivariable industrial processes. Important features required in such a controller are: robust servo and regulatory performance and the ability to handle constraints on the process variables. MOCCA achieves these through a multi-step predictive control strategy. Constraints are handled through on-line optimization procedures such as QP and LP. MOCCA directly employs a non-parsimonious, non-functional step response of the process which may include mild process non-linearities and which is easily generated, as a model. It uses the same general principles as industrially successful schemes such as DMC, MPHC or IDCOM and MAC. However, the main contribution of the present work relative to the original DMC and MPHC publications is a more general formulation, use of a forecast model in the feedback path to enhance the regulatory performance plus the development of more efficient computer solution techniques to solve the on-line optimization problem.

Handling time-delays in multivariable systems requires a suitable definition of what characterizes the "delay structure" of the system. It turns out that the interactor matrix introduced by Wolovich and Falb (1976) characterizes the "delay structure" in MIMO systems and generalizes the SISO concept of delay. Once an appropriate "delay structure" for MIMO systems is defined it becomes rather easy to develop design methods for developing control schemes. In Chapter 4 the role of the interactor matrix in the control of discrete MIMO processes with time-delays is investigated. In particular the problems of feedback control with time-delay compensation using the Smith predictor structure for multivariable systems; feedforward control; model following control and one-step-ahead predictive control are discussed. The multivariable Smith predictor based on the interactor factorization is compared with the recently proposed methods of Ogunnaike and Ray (1979) and Jerome and Ray (1986). The interactor matrix approach turns out to be more formal and optimal in the sense that it factors out the minimum number of delays for each input/output pair.

Parameter Estimation and Adaptive Control: Industrial processes are both complex to model and are time-varying and/or non-linear. Performance of conventional control systems based on fixed time-invariant linear models degrades significantly when the deviations from the nominal operating conditions around which the models are developed are large. Therefore there is a definite need for control systems that can adapt themselves to changing operating conditions and to the time-variations in the plant. Adaptive (or self-tuning) control is often suggested when plant models are not available *a priori* or when the plant is non-linear and/or time-varying. Such a scheme is usually based on estimating the parameters of a plant model or of a control law on-line at each time instant. Several self-tuning control schemes are in existence such as GMV, GPP, LQG and GPC. They range from methods based on minimization of simple one-step cost functions to multi-step cost functions with several possibilities. A new Multi-step Adaptive Predictive Controller (MAPC) has been developed in this thesis based on a state-space model of the process subjected to disturbances and a Kalman filter prediction of output. A major difference between this approach and those existing in the literature is that the state-space model is augmented to allow separate modeling of process dynamics and disturbances. The Kalman filter prediction optimally rejects measurement noise and is a very useful practical feature in industrial environments.

The heart of any adaptive control scheme is the model identification step. It is now well understood that for adaptive control to be successful in practice this step must be done very carefully. This gives rise to the development of improved identification techniques. In Chapter 5 an improved least squares identification algorithm has been presented. This arose out of problems that are encountered with standard recursive least squares (RLS) estimation algorithm. The adaptive control algorithm (i.e. MAPC) proposed in this thesis uses the ILS algorithm for parameter estimation. The design and formulation of MAPC is presented in Chapter 6.

The overall conclusions and some recommendations for possible future work are summarized in Chapter 7.

References

Sripada, N. R. (1986): "A Selected List of Annotated References on Artificial Intelligence and Knowledge Based Systems", Internal Report, Department of Chemical Engineering, University of Alberta, Edmonton, Canada.

Sripada, N. R., D. G. Fisher and A. J. Morris (1985): "Applications of Expert Systems to Process Problems", Energy Processing/Canada, 26, 11.

2. An AI application for process regulation and servo control

2.1 Introduction

Artificial Intelligence (AI) as used in this chapter refers to a broad range of tools, techniques and theory that permits the *symbolic* representation and manipulation of knowledge. AI thus makes it possible to systematically represent and implement many heuristic (or "human") procedures that are simply not practical with conventional computer languages and numerical techniques. Because of its generality, power and flexibility, AI is having, and will continue to have, a profound effect in many areas including process control(Pessel (1986) and Rajaram (1986)). The basic AI techniques are documented in textbooks(e.g. Winston (1984)), and magazines dedicated to "Expert Systems" (e.g. Expert Systems User, IEEE Expert and IEEE Transactions on Pattern Analysis and Machine Intelligence) document some of the current developments and applications.

In the *process control* area there has already been a number of commercially successful AI process control systems, for example on cement kilns (Norman (1985), Tauten (1985) and Zadeh (1984)) but most of the papers to date have been presented at meetings rather than in widely distributed control journals. Some of the early work on fuzzy or AI algorithms for control of dynamic plants was done by Mamdani and others (Gupta *et al.* (1977), Mamdani (1974) and Tong (1977)). However, it is only recently that the process

---

A version of this chapter has been published: N. R. Sripada, D. G. Fisher and A. J. Morris, 1987, IEE Proceedings, 134, pt. D, 4, 251.

control groups in industrial and academic institutions have begun to put a significant effort into AI applications. The purpose of the work described in this chapter is therefore twofold:

1) to demonstrate how AI can be applied to a simple process control problem and to compare the experimental performance versus conventional control techniques;
2) to identify the particular features and concepts that characterize a real-time, sensor-based AI application in an area such as process control.

An earlier overview of AI applications was presented in Sripada *et al.* (1985). The two experiments discussed next illustrate the use of AI in real-time control applications. The AI controllers were implemented in PROLOG running on an IBM personal computer. The experimental process used was an electrically heated thermal capacitance with a variable power supply and a temperature sensor.

2.1.1 Description of the experimental equipment

Figure 2.1 is a schematic of the experimental equipment. The output of the temperature transmitter (TT) conditioned by its associated amplifier/filter is sent to a microprocessor based industrial PID controller from Turnbull Control Systems (TCS) operating in DDC mode. The PC communicates with the TCS controller through the serial, asynchronous port of the PC. The variable power supply is driven by a 4-20 mA signal from the controller and regulates the power to the electrical heater. The process is essentially first order and self regulating but the variable

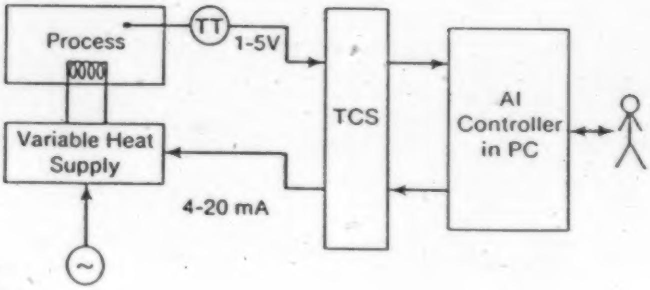

Process
TT
1-5V
TCS
AI Controller in PC
Variable Heat Supply
4-20 mA

Figure 2.1 Schematic diagram of the equipment used for experimental evaluation of AI based controllers

power supply is quite non-linear and presents some control difficulties. The equipment is therefore typical of a simple industrial application that could be used to demonstrate AI techniques.

2.1.2 Data acquisition

The PC based PROLOG system acquires data from the process at successive intervals via the TCS controller. The library of built-in predicates used by the PROLOG system was extended to perform the I/O operations (i.e. reading the measurements and sending the control action) more conveniently. The communication protocol procedures used by the TCS controller were implemented in FORTRAN. The data acquisition takes place at an almost uniform rate, with the time between the samples being determined by the decision time of the AI controllers (typically 2 to 4 seconds).

2.1.3 Knowledge acquisition

The rule base used in the controllers was developed incrementally from the experience of the authors. It was modified iteratively after experimental evaluation.

2.1.4 Experimental procedure

Two sets of experiments were conducted to demonstrate the AI servo and regulatory controllers. Normally both these functions would be combined in a single controller. However, they were implemented separately here for convenience and to shorten the computation time.

2.2 Servo control

The control objective is to drive the process output from an initial steady state to a final steady state in minimal time (i.e. open-loop, optimal state driving). If an exact mathematical model of the process were available then this objective could be achieved using optimal control methods. The resulting time-optimal control policy (called a bang-bang control policy) involves switching the control input alternatively from one extreme to another at precalculated switching times. It is a well known result in optimal control theory that an n^{th} order process would require $\leq n$ switchings of the control input (this does not include the switching of the input to its final steady state) to drive the process exactly to rest at the new steady state. However the precalculated switching times are very sensitive to modeling errors, and also would not work well if noise or disturbances were present. Since an exact model is nearly impossible to obtain in practice, and since noise and disturbances are invariably present in real situations a practical implementation of the optimal control policy always requires some on-line adjustment of the switching times.

A second order time-optimal policy involving two control input switchings between the extremes is often adequate and can handle most practical cases. Figure 2.2 shows a typical situation when a positive setpoint change occurs. The control input is switched from its current value, u_0, to u_{max} when the setpoint change is made and kept at this value until the error $e(k)$ $(:=y_{sp}(k)-y(k)) \leq e^*$. The input is then switched to u_{min} and maintained there for N

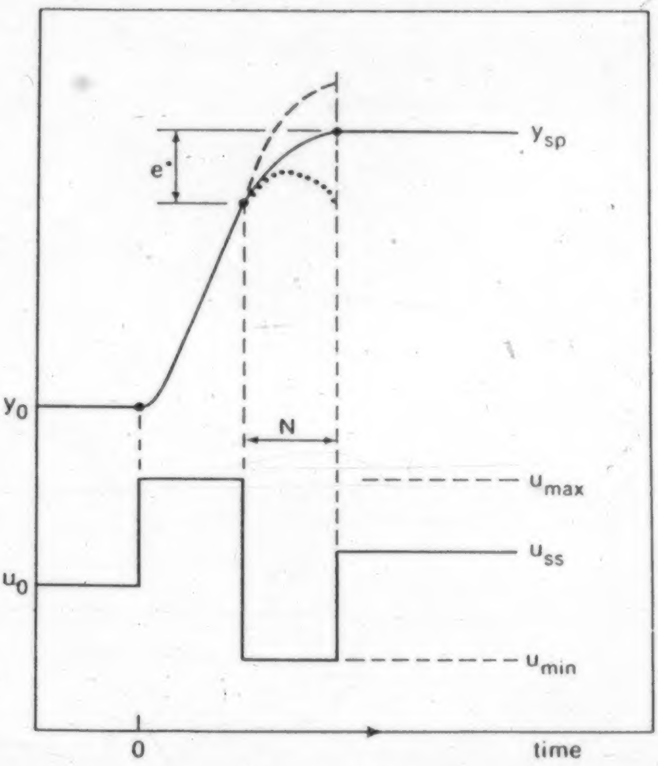

e*
y_sp
y_0
N
u_max
u_ss
u_0
u_min
0
time

Figure 2.2 Time-optimal, open-loop state driving of the controlled variable, $y(k)$, from one setpoint to another showing the effect of errors in e^*

sampling intervals, at the end of which the input is set to its new steady state value u_{ss} (corresponding to the new setpoint). If the switching parameters e^* and N were correct, the output would exactly settle at the new setpoint as indicated by the solid curve in Figure 2.2. If the parameters were wrong, then they could be adjusted heuristically based on the observed response (Nieman (1971)). For example, if e^* were too small then the output would overshoot the setpoint. If e^* were too large then the output would undershoot the setpoint. Similarly, if N were incorrect then the output would still have some momentum (i.e. $dy/dt \neq 0$) by the time the input is returned to its new steady state and exhibit some transient before settling down at the new setpoint.

Since the "optimal" control policy is so easily defined, it is therefore obvious that an AI knowledge based system could be used to implement the control policy and to learn and/or adjust the switching parameters on-line.

2.2.1 Knowledge base

An AI controller was used on the experimental process to implement the second order, open-loop, time-optimal control policy. The knowledge base consisted of two sets of rules (or two knowledge sources):

1) one set ~~of rules~~ for control input switching;
2) one set of rules for learning and/or adjusting the switching parameters.

The rules for control switching specify the direction and time of switching. For example, for a positive setpoint change, the following (informally stated) rules apply:

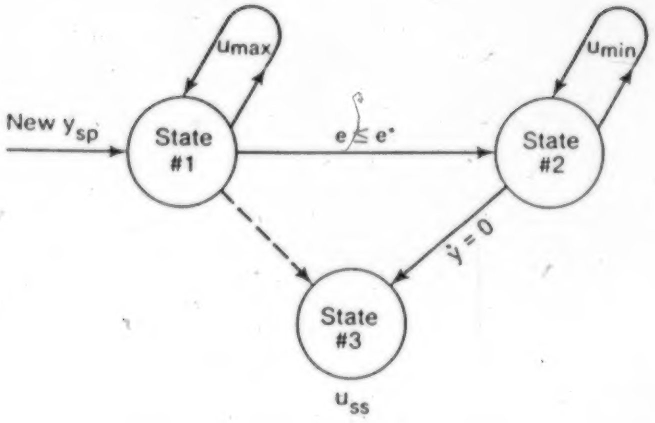
1) make $u(k) = u_{max}$ until $e(k) \leq e^*$;

2) make $u(k) = u_{min}$ for N sampling intervals;

3) make $u(k) = u_{ss}$ where u_{ss} is the new steady state value of the input ($u_{ss} = K_p^{-1} y_{sp}$ where K_p is known or estimated).

The rules for a negative setpoint change are similar. This knowledge source becomes active whenever a setpoint change occurs. For a positive setpoint change, rules (1), (2) and (3) above are executed (or fired) in that order. Note that in this case the time sequence (or order) in which the rules are fired is important. Clearly this knowledge source has a procedural interpretation, because it specifies a procedure (or sequence of actions) to be performed. Representing such procedural knowledge using if-then rules is not straight forward and the resulting set of rules loses much of its transparency. State transition diagrams offer one way of specifying the procedural control. Figure 2.3 is an example of a state diagram which specifies the sequence of actions to take for a positive setpoint change. When a new setpoint is received state 1 becomes active and $u \rightarrow u_{max}$. When $e \leq e^*$ there is a transition from state 1 to state 2 and $u \rightarrow u_{min}$ until the derivative of y becomes zero (i.e. the response has leveled off) at which point the process changes to state 3 and $u \rightarrow u_{ss}$.

There are three parameters, e^*, N and the process gain K_p, that can be learned and/or adjusted from the observed response for each direction of (i.e. positive or negative) setpoint change. The knowledge source for parameter adjustment, which is activated for one cycle at the end of each time-optimal implementation of a setpoint change,



New y_{sp}
State #1
u_{max}
$e \le e^*$
State #2
u_{min}
$\dot{y} = 0$
State #3
u_{ss}

Figure 2.3 State transition diagram representation of the procedure for optimal servo control

specifies how the parameters can be improved based on the observed response data. For example, the rules for adjusting e^* for a positive setpoint change can be informally stated as follows (cf. Figure 2.2):

1) If y overshoots y_{sp} Then e^* is too large and $e^*(k) = e^*(k-1) - \Delta e$;
2) If y undershoots y_{sp} Then e^* is too small and $e^*(k) = e^*(k-1) + \Delta e$.

where "y overshoots y_{sp}", "y undershoots y_{sp}", "e^* is too small" and "e^* is too large" are fuzzy propositions (cf. fuzzy subsets) and are characterized by user specified membership functions. These membership functions and the actual values of y and y_{sp} are used to determine the step size Δe. For example, if "y overshoots y_{sp}" has a membership value of 0.6 then "e^* is too large" is taken to have a membership value of 0.6, which can be converted in to a number for Δe for use in adjusting e^*. (Fuzzy subsets are discussed in Mamdani (1974), Zadeh (1984) and section 2.3 of this chapter)

2.2.2 Discussion of results

Some results obtained from the open-loop, optimal state driving experiments are presented in Figures 2.4 and 2.5. These results consist of a sequence of responses in which the process output y is driven from one steady state to another. The same initial and final steady states were used in all the runs. These results basically illustrate how the AI servo controller learns the correct switching parameters (i.e. e^* and N) starting from an initial set of values.

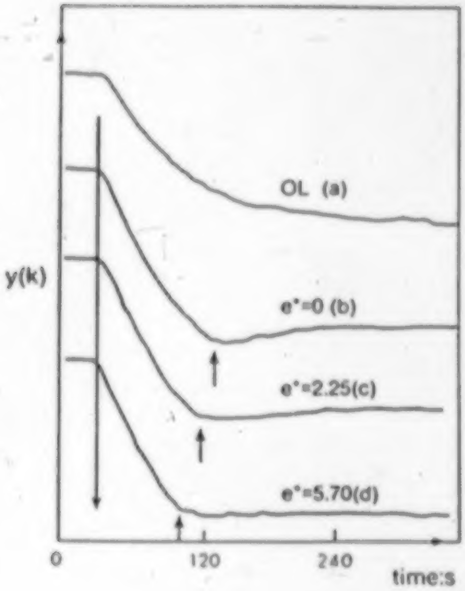


Figure 2.4 A set of experimental runs illustrating how e^* is learned on-line and the corresponding improvement in performance

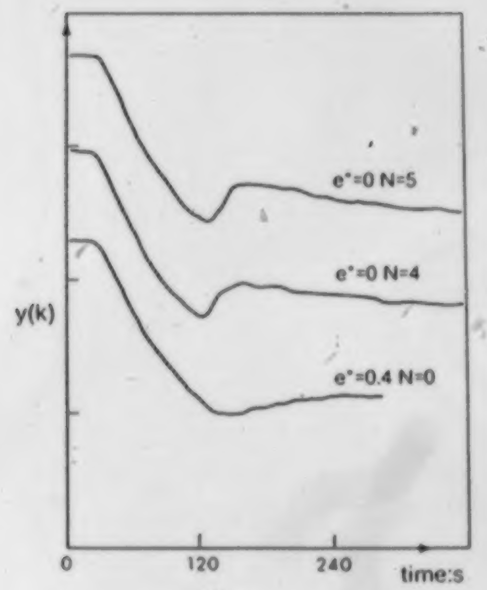


Figure 2.5 A set of experimental runs showing how N and e^* are learned on-line

In Figure 2.4, responses from 4 runs are presented. The response (a) is due to a step change in the input u from its initial value, u_0, to u_{ss}. The AI controller starts with an initial set of switching parameters $e^*=0$ and $N=0$ (response (b)). The overshoot indicates that e^* is too small. For the next run (response (c)) the AI controller uses a larger value of e^* (=2.25) and N=0. The response is definitely better, but still exhibits a slight overshoot. The next run (response (d)), however, is almost optimal and thus the AI controller has learned the correct switching parameters in three steps.

In Figure 2.5, three more time-optimal responses are presented using a different set of initial parameters, $e^*=0$ and N=5 (corresponding to response (a)). Since the process is essentially first order, there is only one switching of the input in the time-optimal policy before returning the input to its new steady state (i.e. N=0, $e^* > 0$). The AI controller learns this by the third run. At this point, the situation is similar to the one corresponding to response (b) in Figure 2.4, i.e. only e^* has to be adjusted.

A measure of the advantage of the optimal controller is obtained by comparing the open-loop AI response in Figure 2.4(d) with that obtained in closed-loop with a conventional, well-tuned PI(D) controller. Figure 2.6 shows the response obtained with a PI controller. The controller gains were set to relatively large values to ensure a fast setpoint change. However, this makes the system sensitive to noise as indicated by the manipulated variable, u(k), in Figure 2.6. The "optimal" response of y(k) is better, i.e. faster and with less overshoot than the PI controller. The

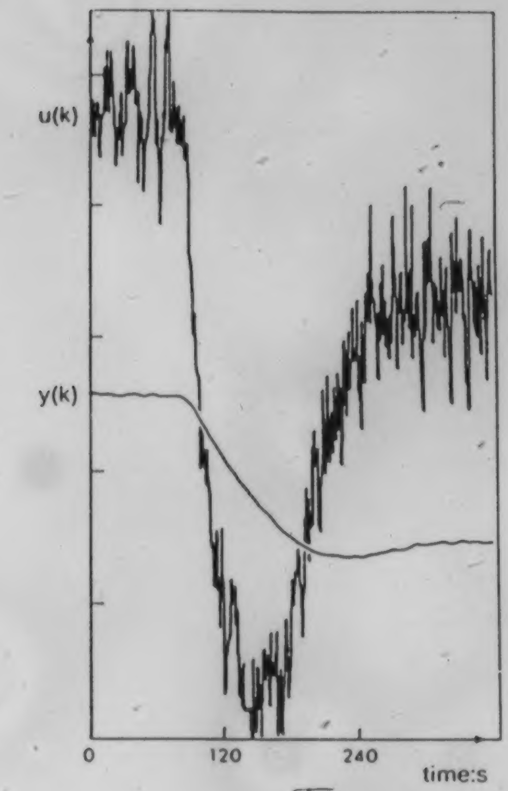


Figure 2.6 Experimental response of the process to a setpoint change using a conventional, fixed-gain PI controller

rise time of the system using PI control could be reduced by increasing the controller gains. However, this results in increased overshoot.

2.3 Regulatory control

"Servo Control" takes a process from its initial value to the desired setpoint conditions as illustrated in the preceding section. "Regulatory Control" compensates for any disturbances that would tend to cause the controlled variable to deviate from the desired setpoint. The objectives of the regulatory controller used in this study were:

1) to remove any significant errors in the output $y(k)$ by appropriate adjustment of $u(k)$;
2) to prevent the process output from exceeding a user specified constraint (i.e. to ensure $y(k) \le y_c \; \forall k$);
3) to produce smooth control action near the setpoint (i.e. minor fluctuations in the process output are not passed to the controller output).

Figure 2.7 illustrates the type of control action desired when the process output is in different regions around the setpoint. Although these control objectives are typical of many practical industrial applications, conventional PID control cannot handle all these objectives, unless extended by additional logic. An experienced human operator can however easily meet all three control objectives. Therefore, to achieve these objectives, an AI controller employing the heuristic knowledge typical of an expert human operator was used.

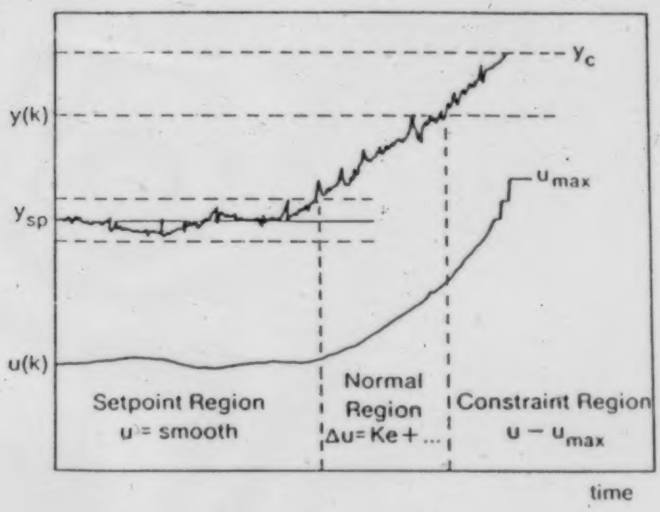

y_c
y(k)
u_{max}
y_{sp}
u(k)
Setpoint Region
u = smooth
Normal Region
Δu=Ke+...
Constraint Region
$u - u_{max}$
time

Figure 2.7 Qualitative representation of the three different operating regions for the AI based regulatory controller

A block diagram of the process and the feedback regulatory controller (AI or PID) is presented in Figure 2.8. A PI controller uses an analytical expression of the following form to compute the control action

$$\Delta u(k) = u(k) - u(k-1) = K_c \left[\Delta e(k) + \frac{\Delta t}{T_i} e(k)\right] \qquad (1)$$

where Δt is the sampling interval and $e(k) = y_{sp}(k) - y(k)$. The control objectives listed earlier would require a variable gain, K_c, in the controller (e.g. a small gain near the setpoint, a very large gain near the constraint etc). Thus a simple PI(D) controller is inherently incapable of meeting all the regulatory control objectives.

The AI controller employs heuristic rules of the form, "If <Situation> Then <Action>" plus fuzzy logic, to deduce the control action. The <Situation> part of each rule is a conjunction of fuzzy propositions, describing a state of the process in which the rule applies; the <Action> part specifies an appropriate change in the control input (e.g. a small negative change) corresponding to that state. Zadeh (1984) illustrates how AI can be combined with fuzzy logic for the control of a cement kiln. Francis and Leitch (1984) describe the use of AI for a simple level control process.

The current state of the process can be adequately described from a knowledge of the current setpoint, value of the constraint (if any) and a sequence of current and past measurements of the process output. The current control error e(k) and the rate of change of output (i.e. dy/dt) were used in the state description. The slope information was obtained by fitting a second order polynomial to the

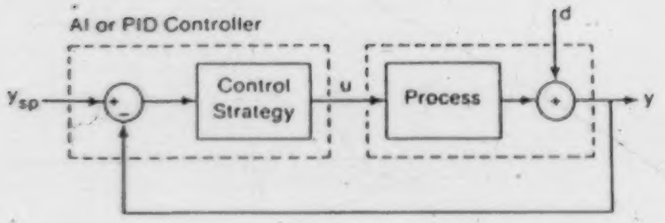

AI or PID Controller
y_{sp}
+
−
Control Strategy
u
Process
d
+
y

Figure 2.8 Block diagram of a standard feedback control system

three most recent output measurements.

In this study, six fuzzy subsets on the universe of discourse of *errors* were used: error is zero (ZE(y)), error is small positive (SPE(y)), error is small negative (SNE(y)), error is large positive (LPE(y)), error is large negative (LNE(y)) and error is in the constraint region (CONSTRAINT(y)). Three fuzzy subsets were used to describe the slope: slope is small (SMALL(dy/dt)), slope is large positive (LP(dy/dt)) and slope is large negative (LN(dy/dt)).

The *incremental* control action was defined in terms of six fuzzy subsets: zero change in the input (ZC(u)), small positive change in the input(SPC(u)), small negative change in the input (SNC(u)), large positive change in the input (LPC(u)), large negative change in the input(LNC(u)) and drastic change in the input (DC(u)). For reasons of simplicity plus limits of computation, only linear membership functions were used to define all the different fuzzy subsets. Figures 2.9, 2.10 and 2.11 show the fuzzy subsets used on their universes of discourse (i.e. e, dy/dt and Δu) all of which in the present case happen to be the real line (R). These membership functions are the prime "tuning variables" when the control engineer installs the AI controller on a new application. Note that the location of the "constraint" membership function in Figure 2.9 moves when the value of y_{sp} or y_c is changed.

2.3.1 Knowledge base

The knowledge base used in the AI regulatory controller is divided into two groups of rules:

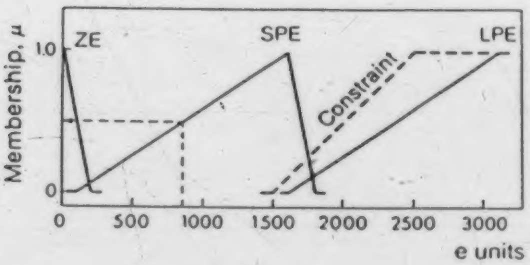


Figure 2.9 Fuzzy set membership functions for the magnitude of the error $e:=y_{sp}-y$ (e.g. $e=850$ units - $\mu_{SPE}(y)=0.5$). The membership functions are symmetric about vertical axis.

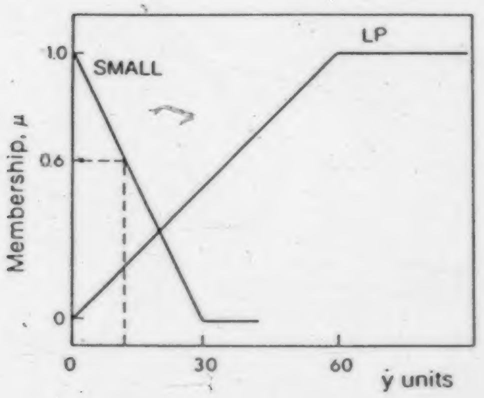


Figure 2.10 Fuzzy set membership functions for the magnitude of the slope of the response (e.g. $y=12$ units - $\mu_{SMALL}(y)=0.6$). The membership functions are symmetric about vertical axis.

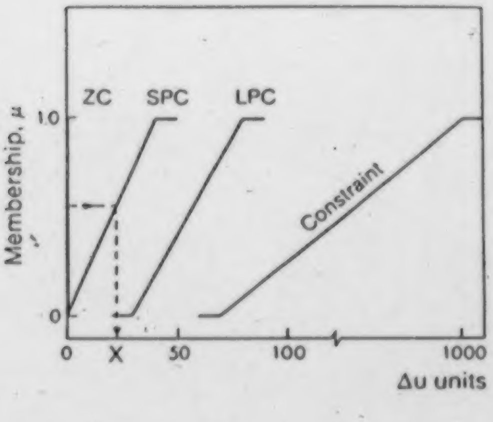


Figure 2.11 Fuzzy set membership functions defining the magnitude of the changes in the manipulated variable (e.g. $\mu_{SPC}(\Delta u)=0.5 = \Delta u=22$ units). The membership functions are symmetric about vertical axis.

1) one group of rules which is always active (i.e. the control action determined from ≥ 1 of these rules is applied every control interval);
2) one group of rules which becomes active only when the process output is near the constraint; the control action from this group is added to or subtracted from group 1.

This partitioning of the knowledge base is necessary to minimize the size of the active rulebase and thereby ensure that the execution time is kept to a minimum. In control applications the AI algorithm is executed every Δt and there is a maximum control interval which if exceeded, will cause a significant reduction in control performance. In this application the desired $\Delta t=2$ seconds and this was exceeded even with the partitioning of the knowledge base.

The group of rules which is always active is summarized below:

1) If ZE(y) Then ZC(u);
2) If SPE(y) and SMALL(dy/dt) Then SPC(u);
3) If SPE(y) and LP(dy/dt) Then ZC(u);
4) If SPE(y) and LN(dy/dt) Then SPC(u);
5) If SNE(y) and SMALL(dy/dt) Then SNC(u);
6) If SNE(y) and LP(dy/dt) Then SNC(u);
7) If SNE(y) and LN(dy/dt) Then ZC(u);
8) If LPE(y) and SMALL(dy/dt) Then LPC(u);
9) If LPE(y) and LP(dy/dt) Then SPC(u);
10) If LPE(y) and LN(dy/dt) Then LPC(u);
11) If LNE(y) and SMALL(dy/dt) Then LNC(u);
12) If LNE(y) and LP(dy/dt) Then LNC(u);
13) If LNE(y) and LN(dy/dt) Then SNC(u).

There are three rules in the group that becomes active when the constraint is approached. One rule specifies the amount of additional control action to be added when in the constraint region:

If CONSTRAINT(y) **Then** DC(u).

The remaining two rules keep track of when the process output approached the constraint region, when it left the constraint region and the total control action generated when the output was in the constraint region. These rules can be stated informally as follows:

If y(k) enters the constraint region **Then** start summing up the constraint control action;

If y(k) leaves the constraint region **Then** subtract the total net control action generated by the constraint rules.

A rule base of this type is necessary to prevent the strong control action taken while the controlled variable y was in the constraint region from dominating the control action in the "normal" region. It is similar to reset windup protection.

Table 2.1 presents a summary of the steps taken by the inference engine of the AI controller at each sampling instance, to compute the control action. For example, if the error e(k)=850 units, then from Figure 2.9 it is concluded that the error in y(k) is "small positive" with a membership value of 0.5. Suppose the rate of change of y, dy/dt (or **y**), is "small" with a membership value of 0.6, then rule 2 is used to deduce that there should be a "small positive change" in u with a membership value of 0.5. Note that the membership value of the <Action> part of a rule is taken as the combined membership value of the <Situation> part. Since

Table 2.1 Summary of the procedure for feedback control using the AI plus fuzzy logic approach

1. Read output $y(k)$ from process
2. Find μ for error e
    - Small ?
    - Large ?
    - Constraint ?
3. Find μ for y-slope, dy/dt
    - Small ?
    - Large ?
4. Rules imply μ for Δu
    - Zero
    - Small
    - Large
    - Drastic
5. Calculate and implement $\Delta u(k)$

the <Situation> part consists of a conjunction of fuzzy propositions, its combined membership value is the minimum of the membership values of the conjuncts. Thus each active rule in the knowledge base specifies a control action. The final control action to be taken is however based on a suitable combination of these individual changes. For simplicity, in this study we have implemented the control input change with the maximum membership value. For example, among all the Δu's specified by the active rules let Δu have maximum membership in the fuzzy subset "small positive change" with $\mu=0.5$. Then as illustrated by Figure 2.11, if Δu is "small positive" with $\mu=0.5$ then $\Delta u=22$ units and this value is sent to the process.

2.3.2 Discussion of results

The response of the AI regulatory controller near the setpoint is presented in Figure 2.12. As desired, the AI controller is insensitive to small perturbations in the output (around the setpoint) and produces a smooth (constant) control action. Examination of the membership function for ZE(y) in Figure 2.9 shows that this is obtained by using what amounts to a deadband around the setpoint. For comparison, the response due to a well-tuned, fixed-gain PI controller is presented in Figure 2.13. The PI controller employs a high gain to quickly eliminate large errors in the output and thereby ensures that the constraint is not violated. Therefore it is very sensitive to even small perturbations in the output and produces large, unacceptable changes in the control input.

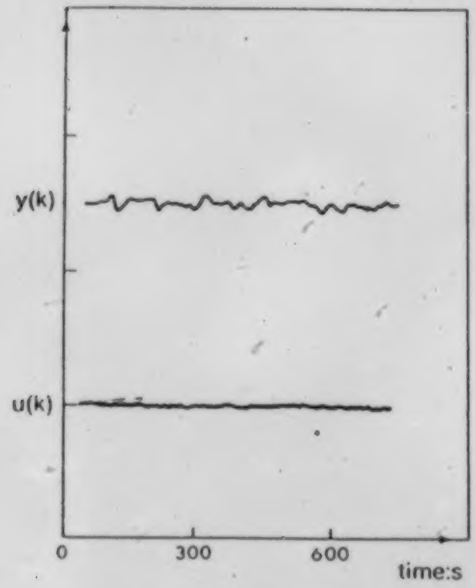


Figure 2.12 Experimental process response near the setpoint using AI control

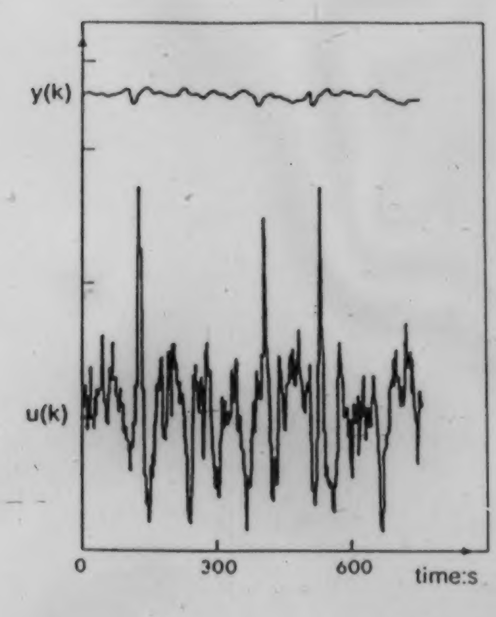


Figure 2.13 Experimental process response near the setpoint using high-gain PI controller

The AI controller can eliminate large errors in the output using vigorous control action as illustrated in Figure 2.14. A step change in load occurred which caused the output to move away from the setpoint. The AI controller generated corrective changes in the manipulated variable u(k) (cf. normal region in Figure 2.7) and $y(k) \rightarrow y_{sp}$. Note that y(k) did not come close to the constraint y_c so the "constraint set of rules" was not activated.

The behaviour of the AI controller when the setpoint is close to a constraint is shown in Figure 2.15. A sufficiently large disturbance occurs which would cause the output to exceed the constraint if special action were not taken. The AI controller, by employing large bursts of additional control action, ensures that the constraint is not exceeded. For comparison the behaviour of a fixed-gain PI controller is presented in Figure 2.16. The PI controller was slightly detuned (i.e. lower controller gain in equation (1)) to make its performance comparable to that of the AI controller near the setpoint and in the intermediate region. The PI controller does not have any knowledge of the constraint and the AI controller makes it possible to operate closer to a constraint than a conventional PI controller. In many industrial applications this translates directly into economic advantages due to improved quantity or quality.

The AI regulatory controller can also handle setpoint changes. However the performance of the AI regulatory controller for setpoint changes must necessarily be worse than that of the time-optimal servo controller discussed earlier. Figure 2.17 shows a typical response due to the AI

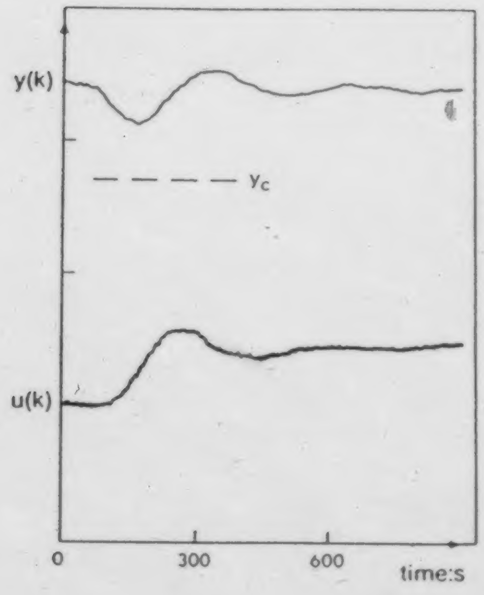


Figure 2.14 Experimental process response for a disturbance using AI control

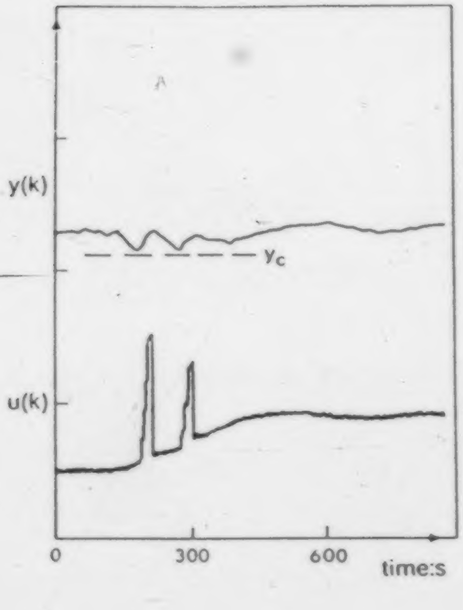


Figure 2.15 Experimental process response for a disturbance using AI control, when a constraint is active

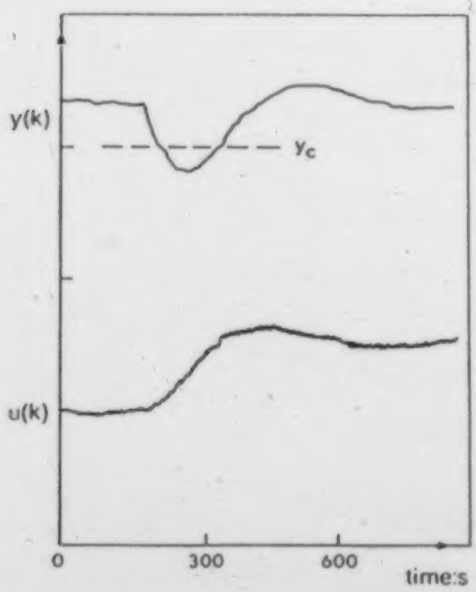


Figure 2.16 Experimental process response for a disturbance using a medium gain PI controller

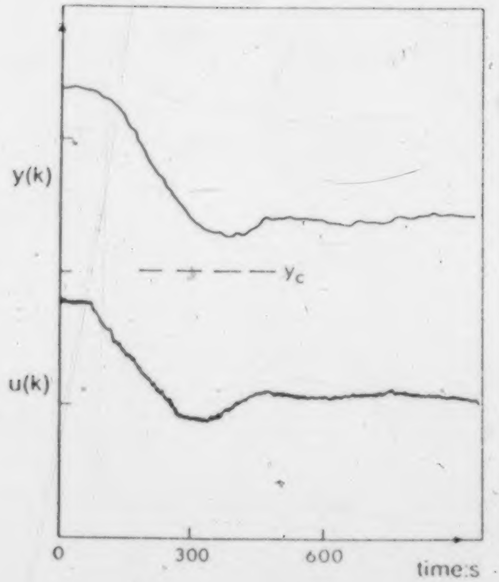


Figure 2.17 Experimental process response for a setpoint change using AI control

regulatory controller for a setpoint change. Figure 2.18 illustrates what happens when a constraint region is approached by the process output during a setpoint change. For comparison, once again the response due to a well-tuned, high-gain PI controller is also shown in Figure 2.19. Note that even though the controlled variable response is good, the control action is unsatisfactory because of its large variance. The AI controllers (regulatory or servo) on the other hand use smoother control action and give a controlled variable response that is as good or better than that given by the widely used PID algorithm.

2.4 Distinguishing characteristics of RTSB applications of AI

Based on the work described in this chapter plus other applications, a number of features that characterize applications of AI in real-time, sensor based (RTSB) process applications have been identified. Some of these characteristics are summarized here for convenient reference.

1) Data are often sensor-based; time-varying; noisy and perhaps qualitative; subject to errors (either bias or complete failure); not always available "on demand", i.e. are available asynchronously to the AI system at times determined by the process; often must be inferred from other data; frequently come from large, on-line databases maintained by conventional process control computers (> 5000 points).
2) Response times are usually important when calculating control action; responding to a process operator's

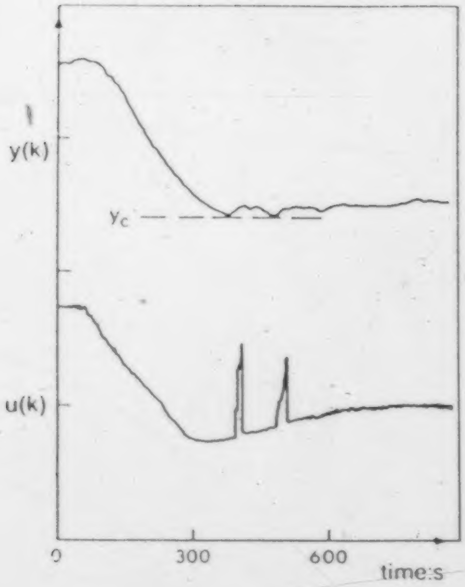


Figure 2.18 Experimental process response for a setpoint change using AI control, when the new setpoint is near a constraint

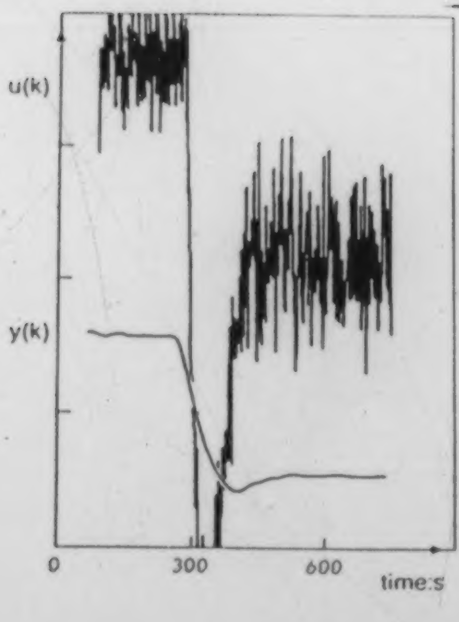


Figure 2.19 Experimental process response for a setpoint change using high-gain PI controller

enquiry; dealing with matters of personnel or equipment safety. For example, if the process response $y(k)$ is rapidly approaching a constraint (cf. Figures 2.15 and 2.18) then the AI controller must respond fast enough to prevent the constraint from being violated.

3) Execution times are important because the AI procedures are executed at intervals as short as a few seconds (Δt=2 to 4 seconds in our applications). Each feedback control loop has a critical Δt which if exceeded will cause a significant deterioration in performance. Similarly it is desireable to handle several control loops at each sampling interval using the same AI-based procedure. Thus execution time is directly related to the number of loops that can be handled and hence to economics.

4) Many RTSB computer system managers like control algorithms to be completely "predictable", i.e. have the same execution time and action for a given set of input data (process state). Many AI systems do not have this property.

5) The control action/decision is implemented directly on the process and hence is not usually subject to human review or directly revocable. Furthermore, the cost of failures or errors can be high and immediate, e.g. offgrade product, damage to equipment, unnecessary process shutdown.

6) The order (time sequence) of data values, states, rules, decisions and/or actions can be important, e.g. they can be procedural rather than simply declarative as illustrated herein.

7) Different sections of the knowledge base may become active or inactive as the process moves to different operating points and/or parts of the physical process are put on-line/off-line. Also the <Action> (or Then) part of a rule may change for similar reasons even when the <Situation> (or If) part of the rule remains unchanged. In this example the results for learning e^* and N only became active after the completion of a setpoint change. Also in the regulatory control, the constraint rules became active only when y approached y_c.

8) Knowledge acquisition is a significant problem. It must often be obtained from a relatively large number of individuals (rather than a single "expert") with significantly different educational levels and understanding of the application, e.g. process operators, engineers, management. The knowledge base is also usually fairly heterogeneous, i.e. consists of simple facts, heuristic rules, deep or theoretical knowledge and data derived from numerical calculations. (The AI controller described in this chapter was simplified and relatively homogeneous.)

9) Process parameters and/or models are usually unknown and must be determined on-line in either qualitative (e.g. fuzzy) or numerical form. This application included the "learning" of the key parameters e^* and N. (cf. Figures 2.4 and 2.5)

2.5 Practical aspects of feedback control

The AI controller discussed in this chapter was never intended as a replacement for the familiar deterministic, PID controller. It was developed to illustrate the use of AI in real-time applications and to give control engineers some idea of its performance and potential. However, the excellent experimental performance suggests that an *extended* version of this controller could be useful in *special* applications, e.g. when constraints are present and/or when the input data is available only in qualitative terms.

One major difference between this work and process control applications by other investigators (Gupta *et al.*, (1977), Mamdani (1974) and Takagi (1985)) is in the reasoning (inference) procedure. When implementing fuzzy logic control applications (e.g. Gupta *et al.* (1977) and Mamdani (1974)) it is common to use the "compositional rule of inference" for reasoning. This is a very formal approach but not necessarily the easiest to understand or implement. The reasoning approach implemented as part of this project is equivalent to the "certainty factor approach" widely used in rule-based expert systems to handle uncertainty and imprecision (Winston (1984)). Takagi and Sugeno (1985) focus on the development of a mathematical tool to build fuzzy models and present their own reasoning procedure. However, their approach does not appear to have any significant advantage for our type of application. The demonstration application presented in this chapter shows how AI can be applied to the servo, regulatory and constraint control problems familiar to all control engineers. Thus it is hoped that the readers can focus on the AI approach rather than the specific application.

Further work is underway to evaluate a number of extensions and improvements: 1) an improved knowledge base of similar design plus alternative approaches such as frames; 2) means for effectively combining open-loop state driving features with the conventional design for feedback controllers; 3) the advantages and disadvantages of using AI to do "gain scheduling" on a conventional algorithm such as PID rather than using AI to calculate the control action directly; 4) extensions to handle more difficult processes, e.g. underdamped, non-minimum phase, time-delay processes etc.; 5) means to reduce the computational effort that must be done at each control interval, e.g. by rewriting some AI procedures into an equivalent program in PASCAL; 6) handling practical, conventional features such as manual/automatic transfer, windup protection in cascaded loops, high/low select options etc.; 7) including a feedback gain constant, e.g. to multiply the abscissa scale on Figure 2.11, for on-line fine tuning; and 8) multivariable, feedforward and interaction effects.

It is important to note that the AI procedures are not unique. They are as varied as the people who create them. Similarly, AI controllers are not (usually) based on theorems and concepts (such as stability) that are familiar to all control engineers. Therefore they tend to be user and application specific. This is an advantage in the sense that an AI controller can be easily tailored for a specific application problem. Conversely it is unlikely that there will ever be an AI controller as universal as the familiar PID algorithm. Nevertheless the authors are optimistic about the potential of AI applications in the process industry and would like to encourage more applications and comparative evaluations.

2.6 Conclusions

1) Artificial Intelligence opens up significant new application areas in the process industries. Some of the potential benefits are illustrated by the AI-based controllers described in this chapter.
2) Fuzzy logic is a very general approach for dealing with heuristic or qualitative relationships and for handling uncertainty, incompleteness and inexactness in data. It can be advantageously combined with AI as illustrated by this chapter.
3) Experimental results showed that the AI approach to the standard servo, regulatory and constraint control problems resulted in performance that equalled or bettered that of a simple, fixed-gain PID controller. An extension of this approach could be useful for selected commercial applications, e.g. where the measurement data, of something like color or smell, was qualitative.

References

Francis, J. C., and R. R. Leitch (1984): "ARTIFACT: A Real-time Shell for Intelligent Feedback Control", Proc. of BCS Conference on Expert Systems, 157.

Gupta, M. M., G. N. Saridis and B. R. Gaines (eds.) (1977): Fuzzy Automata and Decision Processes, North-Holland.

Mamdani, E. H. (1974): "Application of Fuzzy Algorithms for Control of Simple Dynamic Plant", Proceedings of IEE, 121, 12, 1585.

Myers, W. (1986): "Introduction to Expert Systems", IEEE Expert, 1, 1, 100.

Nieman, R. E. (1971): Ph. D. Thesis, Department of Chemical Engineering, University of Alberta

Norman, P., and S. Naveed (1985): "An Expert System Supervisor for a Rotary Cement Kiln", IEE Symposium on Expert Systems in Process Control, University of Salford, Digest No. 1985/107

Pessel, D. (1986): "From the Editor-in-chief", IEEE Expert, 1, 1, 4.

Rajaram, N. S. (1986): "Artificial Intelligence: Its Impact on the Process Industries", ISA-INTECH, 33, 4, 32.

Sripada, N. R., D. G. Fisher and A. J. Morris (1985): "Application of Expert Systems to Process Problems", Energy Processing/Canada, 26, 11.

Takagi, T. and M. Sugeno (1985): "Fuzzy Identification of Systems and Its Applications to Modeling and Control", IEEE Transactions on Systems, Man, and Cybernetics, SMC-15, 1, 116.

Tauten, J. C. (1985): "A Rule Based Supervisory Control System", IEE Symposium on Expert Systems in Process Control, University of Salford, Digest No. 1985/107

Tong, R. M. (1977): "A Control Engineering Review of Fuzzy Systems", Automatica, 13, 6, 559.

Winston, P. H. (1984): Artificial Intelligence, Addison-Wesley.

Zadeh, L. A. (1984): "Making Computers Think Like People", IEEE Spectrum, 21, 8, 26.

3. Multivariable, Optimal, Constrained Control Algorithm (MOCCA)

3.1 Introduction

The "Multivariable, Optimal, Constrained Control Algorithm (MOCCA)" described in this chapter was developed to meet two major objectives: first, a practical, robust, model-based control algorithm that could be applied to multivariable *industrial* servo and regulatory control problems; second, an algorithm that could be implemented effectively on an industrial digital computer.

To be practical for industrial applications it was concluded that the control algorithm should: utilize easily obtained input/output information rather than complicated or theoretical models; be able to follow *trajectories* generated by higher levels of control e.g. optimization; regulate the process despite measured or unmeasured disturbances; handle hard constraints on the process variables; optimize a user specified performance index; and accommodate practical factors such as transducer and/or sensor failures, etc. To be efficiently implemented on a control computer an algorithm: should be simple computationally; not require excessive memory for storage; be insensitive to roundoff errors (short word length); use solution techniques that are explicit, or at least have guaranteed convergence, as opposed to open-ended, iterative calculations; and use established numerical techniques that are documented,

---

A version of this chapter was presented at an international conference on process modeling and control: N. R. Sripada and D. G. Fisher, 1985, Proc. Intl. Conf. on Industrial Process Modeling and Control, Hangzhou, China.

tested, and not application specific. MOCCA meets these objectives plus a number of others as listed in Table 3.1.

3.2 A functional overview of MOCCA

MOCCA is not the result of a specific area of control theory such as state space systems, multivariable frequency domain techniques, stability analysis, etc. Rather it is a computer-based algorithm developed to meet practical control objectives using effective numerical and software engineering techniques. A schematic diagram showing the structure and information flow of MOCCA is shown in Figure 3.1. The actual physical process has an output vector, y, a control vector, u, and measured or unmeasured disturbances, d. MOCCA can be subdivided into three main parts (cf. Figure 3.1):

1) an open-loop forward path for servo control;
2) a feedback path for regulatory control and model-process mismatch;
3) a supervisory system to generate the desired trajectory and constraints.

Forward path for servo control: The predictive controller generates an incremental control action $\Delta u(t)$ at each time interval, t, such that the output of the process "model" $\{y_m(t+i), i\epsilon[1,P]\}$ follows the desired trajectory $\{y_d(t+i/t_2), i\epsilon[1,S], S\geq P, t_2\leq t\}$ and does not violate any of the user specified constraints, e.g. $u_1 \leq u(t) \leq u_2$, $\forall t$. If the process representation (model) is accurate, then the actual process output trajectory $\{y(t+i), i\epsilon[1,S]\}$ will equal the desired trajectory $\{y_d\}$, i.e. servo control will be perfect. The incremental control action, $\Delta u(t)$, is summed to produce

Table 3.1 MOCCA Features

Process Characteristics
- non-square, MIMO
- open-loop stable
- non-minimum phase
- time-varying gain
- known, variable time-delay(s)
- variable $\Delta u(t)$ (time scale)
- some non-linearities

Process Representation
- non-functional, discrete step response
    - experimental
    - calculated
    - via "experience"
- non-minimal model (cf. robustness)
- normalized perturbation variables
- constraints
    - "hard" e.g. $u(t)<u_2$
    - linear $f(x,y,u,z,\ldots)<f_2$
    - time-varying (cf. supervisory control)

Servo (forward path, predictive) Control
- follows specified TRAJECTORY
    - from operator
    - from supervisory control
- user specified performance index
    - linear (IAE, ITAE, ...)
    - quadratic (ISE, ...)
    - control weighting, $r_u(t)$
    - output weighting, $r_y(t)$
- zero offset (integral action)
- gain scheduling

Regulatory Control
- feedforward control (measured d)
- "feedback" of estimated disturbances
    - filter
    - predicted $y(t+i)$, $i \in [1,P]$
    - adaptive

Supervisory Control
- given y_{sp} calculate desired y_d
    - open-loop (i.e. filter)
    - process model + feedback of y
        - improves robustness
- steady state or dynamic optimization

Solution (Control) Algorithm (cf. Gill et al. 1981)
- quadratic programming (QP)
    - bound constraints
    - guaranteed convergence
- linear programming (LP)
    - general LP
    - explicit calculation of Δu

Special Features
- preview servo-control via simulation
- improved robustness
    - increase prediction horizon, P
    - use filtered feedback of y
- "moving window" for prediction/control
    - faster on-line calculations
    - easier off-line tuning
- accommodates sensor/transducer failures

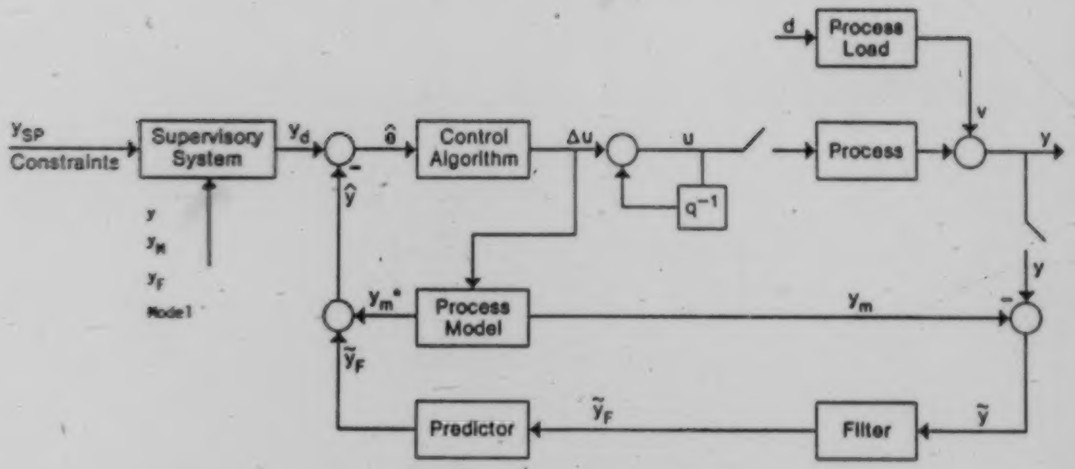

d
Process Load
v
y_SP
Constraints
Supervisory System
y_d
ê
Control Algorithm
Δu
u
q^-1
Process
y
-
ŷ
y
y_M
y_F
Model
y_m*
Process Model
y_m
-
ỹ_F
Predictor
ỹ_F
Filter
ỹ

Figure 3.1 Schematic representation of MOCCA showing process, forward path, feedback path and supervisory system

$u(t)$ before being sent to the process.

Feedback path for regulatory control: The feedback path corrects for the effects of unmeasured disturbances, $d(\cdot)$, and model-process mismatch. The signal $\tilde{y}(t)=y(t)-y_m(t)$ is filtered and then sent to a predictor which produces an estimate of the future values $\{\tilde{y}_F(t+i/t), i\epsilon[1,P]\}$ where P is a user specified design parameter, e.g. P=5. An estimate of the future output trajectory based only on past (known) values $\{\hat{y}(t+i), i\epsilon[1,P]\}$ is then produced by adding $\{\tilde{y}_F\}$ to the model based prediction $\{y_m^*\}$. This estimated trajectory $\{\hat{y}\}$ is then fed back and subtracted from the desired trajectory $\{y_d\}$ to produce the error trajectory $\{\hat{e}(t+i/t), i\epsilon[1,P]\}$, at every control interval rather than a scalar variable such as $e(t)=y_d(t)-y(t)$. Note that the estimate of the output trajectory $\{\hat{y}\}$ could be generated by a number of different filtering and estimation techniques.

Supervisory system: The "supervisory" portion of MOCCA accepts the setpoint trajectory $\{y_{sp}(t+i/t_2), i\epsilon[1,S]\}$ from the process operator (or a higher level computer control) and generates the signal $\{y_d(t+i/t), i\epsilon[1,P]\}$. This system is optional in the sense that y_d could be set equal to y_{sp}. However, the supervisory system can also be used to introduce considerable design flexibility and provides a means to achieve several practical, industrial objectives.

The mathematical formulation for these main components of MOCCA, (the forward servo path, the feedback regulatory path and the supervisory system) will be presented in the next three sections. This will be followed by a brief literature survey and then by a discussion of an evaporator application which illustrates some of the features of MOCCA summarized in Table 3.1.

3.3 MOCCA: Forward path for servo control

The "forward path" in Figure 3.1 could be interpreted as an model inverse controller which, in the ideal case, would cancel the dynamics of the process and result in perfect input tracking, i.e., $y(t)=y_d(t)$, $\forall t$. The theoretical advantages and the practical problems of such a formulation are well known. MOCCA avoids many of these practical problems by using a discrete, non-functional representation of the input/output behaviour of the process and by formulating the objective of $y \rightarrow y_d$ as a constrained optimization problem. Since efficient, often explicit algorithms exist for solving the optimization problem the result is a practical control strategy that can be applied to a broad class of industrial problems.

3.3.1 Model formulation

MOCCA is based on a straightforward, intuitive, time-domain representation of the process and the control objectives. Since this representation is the key to understanding the formulation and implementation of MOCCA it is presented in some detail.

3.3.1.1 Model formulation for SISO processes

The relationship between the input and output variables of the process to be controlled is assumed to be available in the simple, discrete, step response form illustrated in Figure 3.2. That is, for a unit step input at time zero the process output, assuming

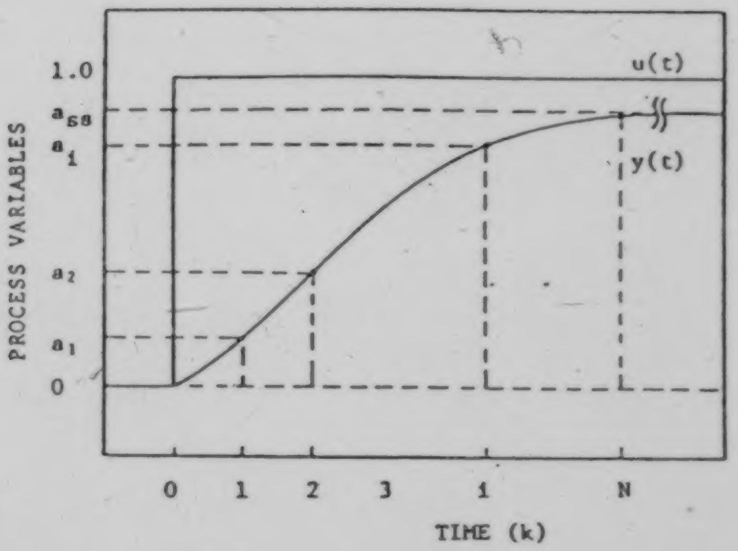


Figure 3.2 Discrete, non-functional representation of the process response, y, to a step change in input; i.e. y(t) is approximated by $\{a_1, ..., a_N\}$

that the process was at zero steady state prior to the application of the step input, is given by:

$$y_m(t+i/0)=a_i \quad i=1, \ldots, N \qquad (1)$$
$$=a_{ss} \quad i>N$$

where N and/or Δt are chosen so that a_N is acceptably close to the final steady state value, a_{ss}. Note that in order to have a finite value of N and a_{ss} the process should be open-loop stable, i.e. no right-half plane poles. (MOCCA can be modified to handle a pole at the origin.) Therefore, unstable processes would have to be stabilized, e.g. by proportional feedback, before MOCCA could be applied. The required process information a_i, $i\epsilon[1,N]$ can be obtained by direct step response testing or by other methods such as deconvolution, correlation etc. Alternatively, the data can be generated by a theoretical model or estimated based on experience and engineering judgement. If the input were a step of magnitude, K_m, then, assuming multiplicative scaling, the a_i and a_{ss} values in equation (1) would simply be multiplied by K_m.

For a single step change of magnitude Δu(t-i) in the input variable at time t-i the change in the process response at time t is given by

$$\Delta y_m(t/t-i)=y_m(t/t-i)-y_m(-\infty/-\infty)=y_m(t/t-i) \qquad (2)$$

if it is assumed that y_m is a normalized deviation variable and that at some time in the past (e.g. at time $t=-\infty$) the process was at steady state so that $y_m(-\infty/-\infty)=0$ when $u(-\infty)=0$. Thus

$$y_m(t/t-i)=a_i \Delta u(t-i) \text{ for } i \le N \qquad (3)$$
$$=a_{ss} \Delta u(t-i) \text{ for } i > N$$

If the input is changed at every previous time interval $t-i$, $i=1, \ldots$ by $\Delta u(t-i)$ where

$$\Delta u(t-i) = u(t-i) - u(t-i-1) \qquad (4)$$

then assuming superposition, the total change in the output $y(t)$ is given by equation (2), or

$$y_m(t/t-1)= \sum_{j=1}^{N} a_j \Delta u(t-j) + a_{ss} \sum_{j=N+1}^{\infty} \Delta u(t-j) \qquad (5)$$

However, since $u(-\infty)=0$, by assumption, the last term can be re-written such that

$$y_m(t/t-1)=\sum_{j=1}^{N} a_j \Delta u(t-j)+ a_{ss} u(t-N-1) \qquad (6)$$

To predict the trajectory $\{y_m(t+i), i\epsilon[1,P], P\le N\}$ it is only necessary to shift the time index in equation (6) such that

$$y_m(t+i/t+i-1) = \sum_{j=1}^{N} a_j \Delta u(t+i-j) +$$
$$a_{ss} u(t-N+i-1),\ i\epsilon[1,P] \qquad (7)$$

At time t the past inputs $u(t-i)$, $i=1, 2, \ldots$ are known and (as shown in the next section) the present and future values of the control changes $\Delta u(t+i-1),\ i \in [1,P]$ can be chosen so that the output trajectory $\{y_m(t+i), i \in [1,P]\}$ has the desired form.

The term in equation (7) involving $\Delta u(\cdot)$ can be partitioned into two parts: the first part involving past input changes and the second involving present and future changes in the input. Then equation (7) can be rearranged as follows to emphasize that the predicted values of the output, are calculated as the sum of two quantities: the contribution of the past inputs plus the contribution from the present and future inputs,

$$\{y_m(t+i/t+i-1), i \in [1,P]\} = \{y_m^*(t+i/t-1), i \in [1,P]\} + A_2\{\Delta u(t+i-1), i \in [1,P]\} \qquad (8)$$

where the contribution of past inputs, $\{y_m^*\}$, is

$$\{y_m^*(t+i/t-1), i \in [1,P]\} = a_{ss}\{u(t-N+i-1), i \in [1,P]\} + A_1\{\Delta u(t-N+i), i \in [1,N-1]\} \qquad (9)$$

$$A_1 = \begin{bmatrix} a_N & a_{N-1} & \cdots & a_2 \\ 0 & a_N & \cdots & a_3 \\ \cdots & & \cdots & \\ 0 & \cdots & a_N \cdots & a_{p+1} \end{bmatrix} \qquad A_2 = \begin{bmatrix} a_1 & 0 & \cdots & 0 \\ a_2 & a_1 & \cdots & 0 \\ \cdots & & & \\ a_p & a_{p-1} & \cdots & a_1 \end{bmatrix}$$

Note that the notation $\{z(i), i \in [1,P]\}$ can be interpreted as a discrete trajectory in the time domain, i.e. $z(1)$, $z(2)$, ... $z(P)$, or as a column vector defined as $(z(1), z(2), ..., z(P))^T$. To implement the control algorithm, the right-hand side of equation (8) is set equal to the desired trajectory $\{y_d(t+i/t), i \epsilon [1,P]\}$ and equation (8) is used as a basis for calculating the present and future control changes $\Delta u(t+i-1), i \epsilon [1,P]$. It might appear easier to assume that the process was at steady state prior to the start of the desired trajectory since the contribution of past inputs given by equation (9) would then be zero. However, the above formulation makes it possible to assume a "moving window" approach to control. Note that the prediction of a future value of the output at time t, e.g. $y(t+P/t)$, is based on the contribution of the preceding N inputs where P of them, $\Delta u(t+i-1)$, $i \epsilon [1,P]$, are the present or future values and N-P of them are known past values. This allows the arbitrary choice of $1 \leq P \leq N$ and hence significant reductions in problem size and computation time.

Matrices A_1 and A_2 are called the dynamic matrices or the internal model of the controlled system since they completely define the dynamic behaviour of the process. In the actual implementation of the control algorithm, considerable savings in computer storage and computation time can be achieved because, for example, it is only necessary to store the vector $\{a_i, i \epsilon [1,N]\}$ rather than the complete A_1 and A_2 matrices.

3.3.1.2 Model formulation for MIMO processes

The relationship between the i^{th}-output and the j^{th}-input variable of a MIMO system can be represented in the same way as for a SISO system. For example a 2x2 MIMO system can be represented (cf. equation (8)) by:

$$\left[\begin{array}{c} \{y_m^1\} \\ \hline \{y_m^2\} \end{array}\right] = \left[\begin{array}{c} \{y_m^{1*}\} \\ \hline \{y_m^{2*}\} \end{array}\right] + \left[\begin{array}{c|c} A_2^{11} & A_2^{12} \\ \hline A_2^{21} & A_2^{22} \end{array}\right] \left[\begin{array}{c} \{\Delta u^1\} \\ \hline \{\Delta u^2\} \end{array}\right] \qquad (10)$$

where by analogy to equation (9)

$$\left[\begin{array}{c} \{y_m^{1*}\} \\ \hline \{y_m^{2*}\} \end{array}\right] = \left[\begin{array}{c|c} a_{ss}^{11}\ I_p & a_{ss}^{12}\ I_p \\ \hline a_{ss}^{21}\ I_p & a_{ss}^{22}\ I_p \end{array}\right] \left[\begin{array}{c} \{u^1(k+i-N-1)\} \\ \hline \{u^2(k+i-N-1)\} \end{array}\right]$$

$$+ \left[\begin{array}{c|c} A_1^{11} & A_1^{12} \\ \hline A_1^{21} & A_1^{22} \end{array}\right] \left[\begin{array}{c} \{\Delta u^1(k-N+i)\} \\ \hline \{\Delta u^2(k-N+i)\} \end{array}\right] \qquad (11)$$

Note that this formulation can accommodate non-square systems and that the parameters P_i and N_i can be different for every pair of input/output variables.

When using equation (10) to solve for the desired control action, $\Delta u(\cdot)$, the most significant dimension is that of the matrix A_2. If all the P_i are equal then A_2 is $(n*P)x(n*P)$ where n is the dimension of the output vector.

3.3.1.3 Model formulation for time-delay systems

SISO systems with a known time-delay of k sample periods are handled in the same way described earlier for non-delayed systems except that the first k coefficients in the step response will be zero since a change $\Delta u(t)$ in the manipulated variable at time t only affects the output for times greater than t+k, (i.e. $y(t+k+i)$, $i \geq 1$). Equation (8) therefore becomes

$$\{y_m(t+k+i/t+k+i-1), i \in [1,P]\} = \{y_m^*(t+k+i/t-1), i \in [1,P]\} + A_2\{\Delta u(t+i-1), i \in [1,P]\} \qquad (12)$$

where equation (9) is modified accordingly and the matrices A_1 and A_2 are the dynamic matrices based on the first N non-zero coefficients of the step response to ensure that A_2 has full-rank, i.e. contains no rows of zeros.

For MIMO processes the procedure is again analogous to the delay free case but care must be taken to ensure that A_2 is of full-rank. A useful rule of thumb is, for each output y_i, to choose the horizon of prediction larger than the largest time-delay, i.e. $\text{maximum}\{k_{ij}, j \in [1,m_u]\}$ where m_u is the number of input variables u_j. Similarly the k_i used on the right hand side of the delayed version of equation (10) should be the minimum$\{k_{ij}, j\epsilon[1,m_u]\}$. For example, for a 2x2 system with delays $k_{11}=2$, $k_{12}=3$, $k_{21}=3$ and $k_{22}=4$, one could choose $P>4$ (or $P_1>3$ and $P_2>4$), $k_1=2$ and $k_2=3$. Then,

$$A_2 = \left[\begin{array}{cccc|ccc}
a^{11}_3 & 0 & 0 & \cdots & 0 & 0 & \cdots \\
a^{11}_4 & a^{11}_3 & 0 & \cdots & a^{12}_4 & 0 & \cdots \\
\cdots & & & & \cdots & & \\
a^{11}_{P_1} & a^{11}_{P_1-1} & & a^{11}_3 & a^{12}_{P_1} & a^{12}_{P_1-1} & \cdots \\
\hline
a^{21}_4 & 0 & & \cdots & 0 & 0 & \cdots \\
a^{21}_5 & a^{21}_4 & & \cdots & a^{22}_5 & 0 & \cdots \\
\cdots & & & & \cdots & & \\
a^{21}_{P_2} & a^{21}_{P_2-1} & & & a^{22}_{P_2} & a^{22}_{P_2-1} & \cdots
\end{array}\right] \tag{13}$$

If a one-step prediction horizon is specified, i.e. $P=M=1$ then an appropriate time-delay factorization scheme, (see Chapter 4) could be used.

It is obvious that a change in any of the time-delays k_{ij} would change A_2 and that the revised value of A_2 would have to be used in all subsequent calculations. Note that to calculate $\Delta u(t)$ it is assumed that all the time-delays are known at time t.

3.3.1.4 Model formulation for measured disturbances

The effect of a measured disturbance, d, on the controlled variable(s) y, can be represented using the same non-functional, discrete, step response relationship that was described earlier for the process input/output relationship (cf. Figure 3.2, equations

(9) and (10)). Future values of the disturbances are of course not measurable. Therefore, $\{d(t+i), i \in [1,P]\}$ can either be set equal to the current value $d(t)$ or can be estimated using one of the filters or single series predictors discussed in the next section on the feedback of y. Once the calculated effect of the disturbance(s) on the process output(s) $\{y_{md}(t+i), i \in [1,P]\}$ is available it can be sent to the input of the predictive controller, i.e. $\{\hat{e}\}=\{y_d\}-\{\hat{y}\}-\{y_{md}\}$. Thus, in effect, feedforward control from measured disturbances can be handled directly by MOCCA. (It is also necessary to add y_{md} to the calculated value of y_m in Figure 3.1 so that $\tilde{y}$ is not influenced by the feedforward control action.)

3.3.2 MOCCA control calculation for servo control

The MOCCA model formulation presented in the previous section showed that future values of the output could be calculated from an equation (cf. equations (8) and (10)) of the form

$$\{y_m\} = \{y_m^*\} + A_2\{\Delta u\} \tag{14}$$

The design objective for the controller in the "forward path" shown in Figure 3.1 is to ensure that the process output approximates the desired trajectory $\{y_d(t+i/t_2), i \in [1,P]\}$. If the process has been at steady state for at least the last N control intervals then the desired control action can be calculated from equation (14).

The trajectory $\{y_m\}$ is simply replaced by the desired trajectory $\{y_d\}$ and since A_2 is square and of full-rank it is possible to calculate the control action $\{\Delta u(t+i-1/t), i \epsilon [1,P]\}$ explicitly:

$$\{\Delta u\} = A_2^{-1} \{y_d\} \tag{15}$$

However, if the process has not been at steady state for at least N intervals then it is necessary to allow for the effect that the past inputs $\{u(t-i), i \epsilon [1,N-1]\}$ will have on the future outputs $\{y(t+i), i \epsilon [1,P]\}$. This is done by calculating the error signal

$$\{\hat{e}\} = \{y_d\} - \{\hat{y}\} \tag{16}$$

i.e. the difference between the desired trajectory and the calculated changes that will occur in the future due to past inputs. If we consider only the "forward path" in Figure 3.1, then $\{\hat{y}\} = \{y_m^*\}$. Thus by using equation (16) to replace $\{y_d\}$ in equation (15), the result is

$$\{\Delta u\} = A_2^{-1} \{\hat{e}\} \tag{17}$$

As long as A_2 is constant then A_2^{-1} is calculated *a priori* and the on-line control calculation is simply the multiplication defined by equation (17). If the model is perfect and there are no disturbances then $y(t)=y_d(t)$, $\forall t$, i.e. the result is perfect model-following or perfect servo control. For P=1 this is equivalent to a perfect one-step-ahead predictive controller. Unfortunately, although the control calculation is simplified, the control performance is often unsatisfactory. For example, the changes {Δu} are often very large and severe oscillations, or ringing in the input are frequently observed.

To be practical for industrial applications the control scheme should:

1) handle process-model mismatch;
2) correct for process disturbances;
3) handle constraints on the process variables;
4) permit design flexibility, e.g. permit different relative weighting on y_i, u_i;
5) be robust in the presence of measurement or modelling errors;
6) allow the user to conveniently specify the performance desired.

Design flexibility can be introduced by noting that the length of the calculated control trajectory $\{\Delta u(t+i/t), i \in [1,M]\}$ can be shorter than the horizon of prediction for the output, P. This means of course that when calculating the control trajectory {Δu} it is not possible to calculate A_2^{-1} analytically as indicated in equation (15). The solution is calculated as indicated by equation (17) where A_2^{-1} is replaced by A_2^{*} which is a pseudo inverse, e.g. $A_2^{*}=(A_2^TA_2)^{-1}A_2^T$. Choosing $M<P$ will give a slower, more robust control system. Also it is typical of MIMO systems that control of one output is more critical than the other outputs and/or that changes, Δu, in a particular input should be reduced due to economic or performance costs. The required design flexibility can be achieved by introducing

weighting on the output or control trajectory e.g. $\{y_j(t+i/t), i\epsilon[1,P]\}$ weighted by $\gamma_{y_{ij}}$, $i\epsilon[1,P]$. For the general case of $M < P$ it is not possible to make y follow y_d exactly. Thus it is desirable to introduce a performance index, J. The performance index can be chosen simply to define the desired control performance. However, it is recommended that the effect of the choice of J on the difficulty of calculating Δu be carefully evaluated.

If the performance index is formulated to minimize the deviations in the (estimated) output $\hat{y}$ from the desired values y_d and to minimize changes in the control variables, then

$$J=\frac{1}{2}[\sum_{i=1}^{P} \gamma_{y_i}\{y_d(t+i)-\hat{y}(t+i/t+j)\}^2+\sum_{i=1}^{M} \{\Delta u(t+i-1)\}^2\gamma_{u_i}] \qquad (18)$$

where $j=\min\{i-1, M-1\}$ and γ_{y_i}, $i\epsilon[1,P]$ and γ_{u_i}, $i\epsilon[1,M\}$ are non-negative weighting factors. However, since $\hat{y}(t+i/t+j)=y_m^*(t+i/t+j)$ and $\hat{y}(t+i/t-1)=y_m^*(t+i/t-1)$ when only the forward path is considered, we can use equations (16) and (8) to obtain the more general, vector-matrix form.

$$J=\frac{1}{2}[\|\{\hat{e}\}-\{A_2\Delta u\}\|_{\Gamma_y}{}^2 + \|\{\Delta u\}\|_{\Gamma_u}{}^2] \qquad (19)$$

where Γ_y and Γ_u are diagonal weighting matrices, e.g. $\Gamma_y=\mathrm{diag}(\gamma_{y_i}, i\epsilon[1,p])$. When feedback is considered $\hat{y}(t+i/t-1)=y_m^*(t+i/t-1)+\tilde{y}(t+i/t)$ as discussed later. The resulting control-law that minimizes J has the same form as equation (17) with A_2^{-1} replaced by A_2^*, where

$$A_2^{*} = (A_2^T \Gamma_y A_2 + \Gamma_u)^{-1} A_2^T \Gamma_y \qquad (20)$$

With this performance index the problem is a standard weighted least squares problem. Note that for most problems A_2^{*} is time-invariant and can be calculated off-line. MOCCA can also be reformulated using a broad range of linear or quadratic performance indices.

Remark

The numerical properties of the matrix inversion step in equations (17) and (20) can be improved significantly by applying optimal scaling discussed in Chapter 5. Essentially this procedure involves inverting an optimally scaled matrix rather than the original matrix. See Chapter 5 for details on how to compute an appropriate scaling matrix.

3.3.2.1 MOCCA with constraints

In many practical problems it is necessary to place constraints of the form $z_1 \le z(k) \le z_2$ where z is an output variable y_i; input variable u_j; state or intermediate variable; derivative of one of the process variables; or any linear function of u, y and/or their derivatives. The result is a constrained optimization problem of the form

minimize J

such that $z_1 \le Gz(t) \le z_2$

where G is an appropriate matrix.

The algorithm used to solve the constrained optimization problem depends on the form of J and of the constraints. It is selected with due consideration to its numerical properties, stability, memory requirements, *a priori* and on-line computation time, plus its rate of convergence and robustness.

For most MOCCA applications with a quadratic performance index (cf. equation (19)) it is possible to use a quadratic programming algorithm such that all the on-line calculations have guaranteed convergence. With some linear formulations the calculations are explicit. QP and explicit calculations are much faster than iterative calculations or general optimization schemes such as the one outlined by Richalet *et al.* (1978). Explicit calculations have the added advantage that their execution time is relatively constant and hence the user can be relatively certain that the control scan interval will not be violated.

3.4 MOCCA: Feedback path for regulatory control

The open-loop MOCCA control strategy described in the previous section would obviously not give satisfactory control in the presence of disturbances or model-process mismatch. Therefore a feedback system is added as shown in Figure 3.1. Assuming superposition is valid, the effect of disturbances such as d(t) on the output y(t) can be represented by a single signal v(t) as shown in Figure 3.1. The signal $\tilde{y}(t)=y(t)-y_m(t)$ is equal to the sum of the disturbance effects v(t), the effect of model-process mismatch, plus other non-idealities. (However, it is easier to understand the design basis of the feedback system for the ideal case of perfect modeling and a single disturbance such that $\tilde{y}(\cdot)=v(\cdot)$).

3.4.1 Feedback interpretation of regulatory control

For feedback control the objective is, given an actual measurement of the process, to construct an accurate estimate of the future output trajectory $\{\hat{y}(t+i/t), i\epsilon[1,P]\}$ that can be used to calculate the error trajectory $\{\hat{e}\}$. There are several filtering and estimation techniques that could be used to generate $\{\hat{y}\}$. Perhaps the simplest is to set $\tilde{y}(t+i/t)=\tilde{y}(t), i\epsilon[1,P]$. However, Figure 3.1 shows a "filter" plus a "predictor" which, based only on present and past values of $\tilde{y}=y-y_m$, will produce an estimate of the future values $\{\tilde{y}(t+i/t), i\epsilon[1,P]\}$. The desired estimate of $\{\hat{y}\}$ is then produced by summing $\{\tilde{y}\}$ and $\{y_m^*\}$. The filter/predictor is discussed more fully later in this section. Note that for linear systems the order of the filter and the predictor could be interchanged in Figure 3.1.

3.4.2 Feedforward interpretation of regulatory control

In conventional control systems that use only the current values of process signals, e.g. $d(t)$, measured disturbances, d, can be compensated by an appropriate feedforward controller. If d is not measured then it is only a slight extension to think of feedforward control based on an estimate of the disturbance, $\hat{d}$. If an estimate of $v(\cdot)$ were used in place of $\hat{d}(\cdot)$ then the feedforward controller would be a model-inverse controller. Thus "feedback" schemes for control systems similar to Figure 3.1 that include inverse model predictive controllers can also be interpreted as compensating for disturbances via feedforward action based on an estimated $\hat{v}(\cdot)=\tilde{y}(\cdot)$. Since MOCCA uses trajectories rather than simply the current value (e.g. $\tilde{y}(t)$), it is necessary to include a predictor in the feedback path which will produce an estimate of $\{\tilde{y}(t+i/t), i \in [1,P]\}$ based on present and past values of $\tilde{y}(\cdot)$. This disturbance predictor is discussed below.

3.4.3 Formulation of the feedback predictor

A unified approach to disturbance modelling and prediction can be developed (Astrom 1984) by using a "disturbance generator" of the form

$$v(t)=\frac{C(q^{-1})}{A(q^{-1})}d(t) \tag{21}$$

where q^{-1} is the backwards shift operator, A and C are polynomials (degree C < degree A, and C is assumed stable) and d(t) is an independent, zero-mean, white-noise sequence or a sequence of well isolated pulses. (A discrete, piece-wise deterministic signal can easily be transformed into the required pulse form so that the representation of d is fairly general). Depending on C and A, v(t) may be a step, ramp, sinusoid or a more complicated signal. Note that this representation is unique and is a convenient means of handling both deterministic and stochastic disturbances. C and A can be determined in a number of ways including from open-loop step response data as used for the MOCCA process input/output model.

Prediction is achieved by defining unique polynomials F and G such that

$$\frac{C(q^{-1})}{A(q^{-1})}d(t+m)=F(q^{-1})d(t+m)+\frac{G(q^{-1})}{A(q^{-1})}d(t) \tag{22}$$

where

$$F(q^{-1})= 1+f_1q^{-1}+ \ldots +f_{m-1}q^{-m+1}$$

$$G(q^{-1})=g_0+g_1q^{-1}+ \ldots +g_{n-1}q^{-n+1},\ n=\deg(A)$$

The m-step ahead prediction of $\hat{v}$ is obtained by setting $d(t+m)=0$ and using equation (21) to substitute for the left hand side of equation (22):

$$\hat{v}(t+m/t)=\frac{G(q^{-1})}{A(q^{-1})}d(t) \tag{23}$$

or by again using equation (21),

$$C(q^{-1})\hat{v}(t+m/t)=G(q^{-1})v(t) \tag{24}$$

In MOCCA $v(\cdot)$ is replaced by $\tilde{y}(\cdot)$ so that model-process mismatch and other non-idealities can be included. Equation (24) can be modified accordingly and re-expressed as

$$\tilde{y}(t+m/t)=[\sum_{i=0}^{r} \omega_i^{m}q^{-i}]\tilde{y}(t) \tag{25}$$

where $\sum_{i=0}^{r} \omega_i^{m}=1$ and ensures that the steady state gain of the feedback path is unity. Here the coefficients $\{\omega_i^{m}\}$ are defined for an m-step-ahead prediction of $\tilde{y}$. Thus for $m\epsilon[1,P]$ we would have P sets of such coefficients. Note that the predictor for both deterministic and stochastic systems is an r^{th}-order polynomial. Also note that the disturbance d(t) is not predicted directly and that equation (25) is a "single series forecaster" rather than an input/output model. The coefficients $\{\omega_i^m\}$ can be obtained from the open-loop response data or estimated on-line. D. Man (1984) has shown that a "single series forecaster" can significantly improve the performance of predictive control systems such as MOCCA.

3.4.4 Feedback filter formulation

In most MOCCA applications the filter is simply selected as unity or a low-pass filter to reduce noise and aliasing. However, it can be designed to enhance closed-loop stability (cf. Garcia and Morari 1982) or to influence how strongly and quickly the controller reacts to disturbances $d \rightarrow v \rightarrow \tilde{y}$. The user also has the option of moving the feedback filtering into the "supervisory system" as described below.

3.5 MOCCA Supervisory system

In the most general case the "supervisory" calculations can be based on input from the process operators or from higher level control programs plus feedback of information such as y, y_m, $\tilde{y}$ etc. Calculations can range from complex optimization or constraint identification to simple filtering of the setpoint y_{sp}. Typical applications are to calculate the time-varying gains and/or constraints used by the MOCCA controller, or to improve the performance of non-minimum phase processes.

One option, which resembles the approach used in MAC, is to filter step inputs in setpoints (which could be generated by a steady state optimization program) such that $\{y_d(t+i/t), i\epsilon[1,P]\}$ exhibits the desired dynamics and $y_d(t)$ always equals the current measured output, $y(t)$. For example, assuming first order dynamics.

$$y_d(t)=y(t)$$

$$y_d(t+i)=\beta y_d(t+i-1) + (1-\beta)y_{sp}(t), \ i>1 \qquad (26)$$

The choice of $\beta\epsilon[0,1]$ influences the process response to both setpoint or load changes and influences the closed-loop stability and robustness.

3.6 MOCCA, DMC and MAC: Literature Survey

MOCCA may be considered as a formalization and extension of the "Dynamic Matrix Controller (DMC)" developed by Shell Oil in the USA. Many of the design concepts are also similar to those used in the "Model Predictive Heuristic Control (MPHC)" or the "Model Algorithmic Control (MAC)". For example, all of these schemes used predictive controllers based on non-parametric step or impulse-response representations of the process input/output behaviour and use some type of optimization algorithm to calculate control action such that $\{y\}\rightarrow\{y_{sp}\}$. However, there are also important differences. Some of the similarities, differences and their implications for industrial applications are discussed below.

Motivated by Brosilow's (1979) interpretation of "Inferential Control", Garcia and Morari (1982) developed their "Internal Model Control (IMC)" structure which showed the relationship between, and generalized the concepts of a wide range of control systems including, "Smith predictors" and "deterministic, model-based, predictive control schemes" such as DMC. This paper is excellent for the perspective it offers but does not deal with the practical details of algorithms such as DMC.

3.6.1 Dynamic Matrix Control (DMC)

DMC has been used for over a decade by Shell Oil, USA, but it was first presented in 1979 by Cutler and Ramaker. It was this paper that motivated the MOCCA project at the University of Alberta. Prett and Gillette (1979) describe a catalytic cracker application in which the setpoints for DMC are generated by a steady state (linear programming) optimization. The DMC included some process constraints and weighting but the implementation was more ad hoc than formal. Cutler (1982) showed that DMC could be applied to "imbalanced heat or material balance systems", e.g. to an open-loop unstable liquid level control process (i.e. an integrator). DMC was also shown to compare very favorably with conventional PID and other controllers. In a very short conference paper Garcia and Morshedi (1984) discuss a revised version of DMC based on the principles of IMC and a more formal quadratic programming approach. Unfortunately, very few details are given.

3.6.2 Model Algorithmic Control (MAC)

The first paper presented by Richalet *et al.* in 1977 and published in 1978 described several successful industrial applications of MAC. Later papers by Mehra *et al.* (1979, 1980a and 1980b) and Reid *et al.* (1982) focused mainly on theoretical considerations. The various publications used several different names to refer to this class of algorithms, e.g. Model Predictive Heuristic Control. MAC is based on three main principles:

1) the process is modelled by its discrete impulse response and outputs are calculated by discrete convolution;
2) the desired closed-loop behaviour is specified by a trajectory; for setpoint changes the trajectory is normally generated by the input of a step change into a user specified model;
3) the control algorithm ranges, depending on the problem formulation, from simple least-squares estimates to a heuristic, three-level, iterative, constrained optimization scheme; details of the formulation and solution algorithm are not published.

The theoretical results are difficult to summarize concisely. Usually they apply to a specific problem formulation and solution algorithm rather than to the generalized MAC system. Mehra *et al.* (1979) show that the robustness and stability of MAC is due to: the redundancy (non-minimal representation) of the process representation; the closed-loop updating of the reference trajectory; and the characteristics of the filter used to generate the reference trajectory. It is also shown that it is possible to use MAC to obtain a state dead-beat controller or to achieve arbitrary pole placement/eigen-structure assignment.

Mehra and Rouhani (1980) consider the theoretical and practical problems that arise with non-minimum phase (NMP) systems. Basically, the process zeros that lie outside the unit circle cause "instability during the inverse model calculations" done by the control algorithm. They suggest overcoming this problem by modifying the impulse response; pole placement; using constant inputs over the NMP portion of the response; output weighting, etc.

3.6.3 Other approaches

There is a growing number of people working on developing or applying algorithms of the same general type as DMC, MAC, and MOCCA. Ogunnaike (1983) pointed out the resemblance of DMC features to some statistical concepts such as Bayesian sequential decisions, Levenberg-Marquardt non-linear regression, etc. Marchetti *et al.* (1983) developed their own algorithm, compared it with other controllers and obtained both simulation and experimental results for a stirred-tank heater.

3.6.4 Industrial applications

DMC was developed by Shell, USA, and has been successfully applied to a number of process applications including distillation columns, catalytic reactors, furnaces, etc. The most significant indicator of the merits of DMC is that after a decade of experience Shell is continuing to develop and apply DMC in new and existing installations. Richalet *et al.* (1978) report that MAC has been continuously and successfully applied to a dozen large scale industrial processes for more than a year's time. Their paper describes applications to a complete poly-vinyl-chloride (PVC) plant a distillation column and the steam generation portion of a 250 MW power plant.

3.7 MOCCA: Evaporator application

The pilot plant evaporator at the University of Alberta has been the object of a number of modelling, identification and control studies over the past 20 years. It therefore provides a convenient means of evaluating MOCCA and comparing it with other controllers. The evaporator (which is shown schematically in Figure 3.3) has been described in a number of previous publications, e.g. Fisher and Seborg, 1976. The objective is to keep the outlet concentration, C2, equal to its desired value despite disturbances such as changes in feed flow rate. It is also necessary to control or constrain the liquid levels in the two effects so that they stay within operating limits. The levels W1 and W2 are usually controlled by manipulating the outlet flows B1 and B2 from each effect. The concentration can be controlled by manipulating the steam flow, S, to the first effect. A linearized fifth-order linear state-space model can be derived from basic material and energy balances and represents the output behaviour reasonably well. A simpler third-order model was used to generate the step response sequences required by MOCCA. However, in the simulation studies the fifth-order model was used to represent the

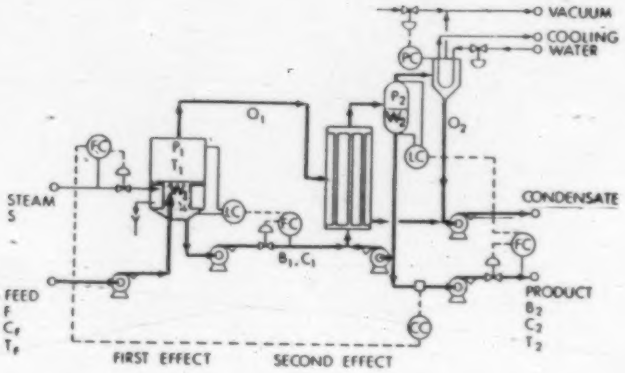

VACUUM
COOLING
WATER
PC
P2
W2
O2
O1
FC
P1
T1
W1
STEAM
S
LC
FC
LC
CONDENSATE
FC
B1, C1
FEED
F
CF
TF
CC
PRODUCT
B2
C2
T2
FIRST EFFECT
SECOND EFFECT

Figure 3.3 Schematic diagram of the double effect evaporator

actual process. Thus there was significant model-process mismatch in all of the evaporator runs described below. The evaporator models contain two open-loop unstable (integrator) modes corresponding to the two levels. These were stabilized by low gain, proportional feedback control before MOCCA was applied. All runs were done using a 20% change in the feed flow rate, F, to the first effect.

3.7.1 MOCCA parameters

For MOCCA applications it is necessary to specify the following design parameters: the control interval, Δt; the length of the open-loop step response sequence, N; the horizon of prediction for the output, P; the number, M, of control values $\{\Delta u(t+i), i \in [1,M]\}$ calculated at each interval; the output weighting, Γ_y; the control weighting, Γ_u; the feedback filter parameter(s), α; and if they are used, the parameters for feedback prediction, $\{\omega_i^m\}$, the constraints, and the supervisory system, e.g. β. It should be noted that when MOCCA is used, a process model (step response data) is necessary and hence is available for process simulation and parameter evaluation.

The control interval is chosen using the usual guidelines for discretizing continuous systems. (However, Δt should not be too small since this would require a large value of N and could result in small values of the step response coefficients $\{a_i\}$ which could lead to numerical difficulties.) For these runs Δt was set to 64 seconds which is the value used in most of the previous evaporator studies. N=40 gave a reasonable truncation error (cf. Fig.

3.2). The rest of the parameters were set equal to typical "default" values and then tuned in a series of simulation runs. Typical default values are P=10, M=5, Γ_y=I, Γ_u=0, α=0, no constraints, filter, prediction or supervisory action.

3.7.2 Evaporator run #1

The prediction horizons, P=5 and M=3 were chosen to show that long prediction horizons are not required; Γ_y=diag(0.1I, 0.1I, 10I) to weight the concentration 100 times more than the levels; Γ_u=diag(0.75I, 0.25I, 0.75I) because steam was a direct expense and B_2 was more critical than B1 from an operating point of view, e.g. large fluctuations in the product flow are undesirable. These weights follow directly from operating experience and the control objectives. (The relative weighting is more important than the actual numerical values.)

The setpoint change in Figure 3.4a shows a good C2 setpoint response with significant deviations in the levels. (All variables are normalized, perturbation variables $Z=(\tilde{Z}-Z_{ss})/Z_{ss}$ so they start at zero.) The control action in Figure 3.4b shows relatively large deviations. In fact B2, goes below -1.0 which indicates a negative flow. From a physical point of view it can be seen that a step change in setpoint, C2, results in: an increase in steam flow, S, to evaporate more water; a reduction in B1 (and hence an increase in W1) to reduce "dilution" effect on C2; and a reduction in B2 to reduce the production of "offgrade" product. (W2 decreases because the effect of B1 and S is stronger than the effect of reducing B2). Note that the

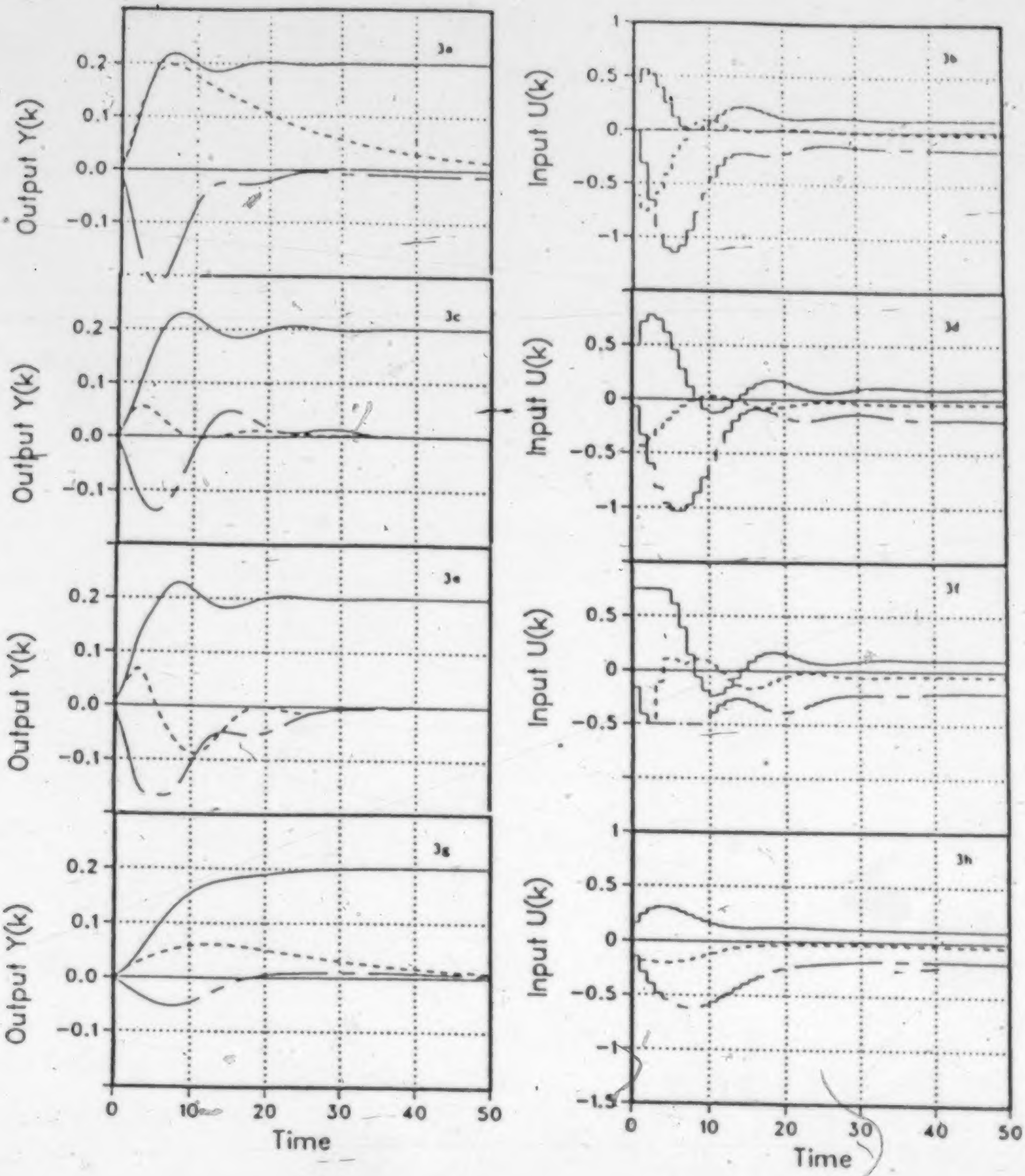


Figure 3.4 Simulated responses of the evaporator to a 20% increase in C2 setpoint; left column: C2, continuous; W1, small dashes; W2, large dashes; right column: S, continuous; B1, small dashes; B2, large dashes

oscillations in C2 are probably due to interactions from B1 rather than too high a gain (weight) on C2. A similar interpretation could be given for each run but will be omitted since the main interest in this paper is on control performance. The magnitude of the overshoot and oscillation in C2 expressed in physical terms is approximately 0.1% glycol compared to a steady state value of 10.0.

The responses of the output variables to a 20% change in feedrate F are plotted in Figure 3.5a. The overshoot in the control variables (Figure 3.5b) is smaller than for the setpoint change and does not violate any physical constraints.

3.7.3 Evaporator run #2

To generate better level control the weighting on W1 and W2 was increased from 0.1 to 1.0. A comparison of Figure 3.4c with 3.4a shows that the desired objective was obtained at the cost of slightly increased oscillation in C2. Figure 3.4d shows that the maximum value of steam flow was approximately 0.75 as compared to 0.5 in Figure 3.4b, i.e. tighter control specifications cause stronger action.

For the 20% load change in F, Figure 3.5c shows that the level deviations are significantly smaller than run #1 but the deviation in C2 is slightly larger. This was expected because of the higher weighting on the levels relative to C2 and to the control action. The control action in Figure 3.5d is also stronger than in 3.5b.

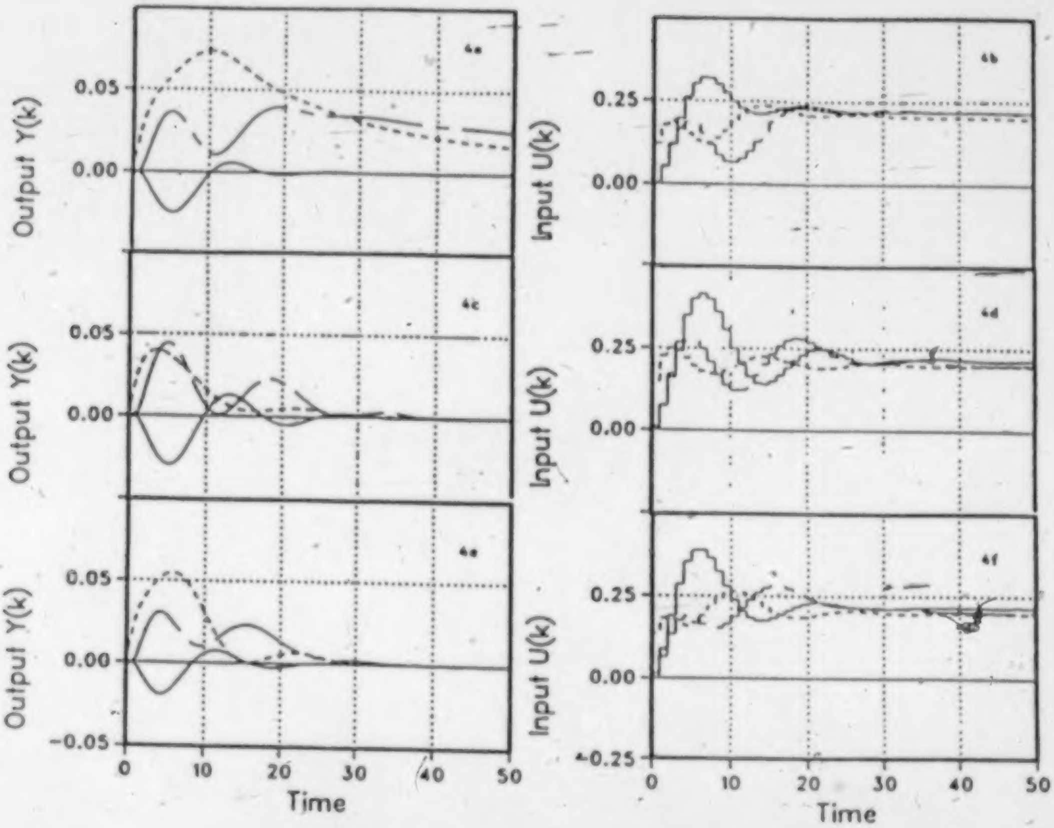


Figure 3.5 Simulated responses of the evaporator to a 20% increase in feed flow rate; left column: C2, continuous; W1, small dashes; W2, large dashes; right column: 5, continuous; B1, small dashes; B2, large dashes

3.7.4 Evaporator run #3

This run illustrates the effect of the following output and control constraints: magnitude of W1≤0.15, W2≤0.15, S≤0.75, B1≤0.5 and B2≤0.5. To compensate for the expected decrease in control quality on the output variables the relative weighting of the output versus the control variables was increased by a factor of 10, i.e. Γ_u=(1,1,10). The resulting C2 response shown in Figure 3.4e is very similar to run #2, but the deviations in the level are larger. (W2 actually hits the constraint at -0.15). The reason for this is clear from Figure 3.4f which shows that all three manipulated variables hit their constraints and hence control of the output variables was not tight. The response to a 20% change in F (Figure 3.5e) shows better control of C2 than in Figure 3.5c because neither B1 nor S reached their constraints (Figure 3.5f), and the weighting on the output variables is higher than run #2. The initial overshoot in W1 is larger in Figure 3.5e than 3.5c which is somewhat surprising.

3.7.5 Evaporator run #4

In this run all the parameters are the same as in run #3 except that a filter with β=0.8 was put in the supervisory system (cf. equation (26)). This has the effect of softening the demand for change, i.e. a step change in y_{sp} or v would reach e as an exponential. The C2 response in Figure 3.4g follows the filtered setpoint almost exactly and the level control is more sluggish than in the previous runs. Since the demands on the system are less the control action in Figure 3.4h is much smoother than in Figure 3.4f and there is a much smaller maximum control action.

3.8 Conclusions

1) MOCCA has a number of features that are important in industrial applications as shown by Table 3.1.
2) The discrete, non-parametric, step response data are a convenient, easy to obtain process representation.
3) The control objectives such as desired output trajectories, constraints, etc. are specified directly in the time domain and hence follow directly from operating goals and experience.
4) Efficient algorithms are available to solve the constrained optimization problem and generate the required control action.
5) MOCCA gives excellent results when applied to practical problems.

References

Astrom, K. J. and B. Wittenmark (1984): *Computer Controlled Systems: Theory and Design*, Prentice-Hall.

Brosilow, C. B. (1979): "The Structure and Design of Smith Predictors from the Viewpoint of Inferential Control", Proc. JACC, 288.

Cutler, C. R. and B. L. Ramaker (1979): "Dynamic Matrix Control - A Computer Control Algorithm", Paper ~~no.~~ 51b, AIChE 86th National Meeting.

Fisher, D. G. and D. E. Seborg (1976): *Multivarible Computer Control: A Case Study*, North-Holland.

Garcia, C. E. and M. Morari (1982): "Internal Model Control. 1: A Unifying Review and Some New Results", I&EC Process Des. Dev., 21, 2, 308.

Garcia, C. E. and A. M. Morshedi (1984): "Solution of the Dynamic Matrix Control Problem via Quadratic Programming (QDMC)", Proc. CICS conference, Ottawa.

Gill, P. E., W. Murray and M. H. Wright (1981): *Practical Optimization*, Academic Press.

Lawson, C. L. and R. J. Hanson (1974): *Solving Least Squares Problems*, Prentice-Hall.

Lee, W. and V. W. Weekman, Jr. (1978): "Advanced Control Practice in the Chemical Process Industry: A Review from Industry", AIChE J., 22, 1, 27.

Maarleveld, A. and J. E. Rijnsdorp (1970): "Constraint Control on Distillation Columns", Automatica, 6, 51.

Man, D. (1984): "Process Control using Single Series Forecasting", M. Sc. Thesis, Depart. of Chemical Engineering, Univ. of Alberta.

Marchetti, J. L., D. A. Mellichamp and D. E. Seborg (1983): "Predictive Control Based on Discrete Convolution Models", I&EC Process Des. Dev., 22, 488.

Mehra, R. K., R. Rouhani, A. Rault and J. G. Reid (1979): "Model Algorithmic Control: Theoretical Results on Robustness", Proc. JACC, 387.

Mehra, R. K. and R. Rouhani (1980a): "Theoretical Considerations on Model Algorithmic Control for Non-minimum Phase Systems", Proc. JACC, TA8-B.

Mehra, R. K., R. Rouhani and L. Praly (1980b): "New Theoretical Developments in Multivariable Predictive Algorithmic Control", Proc. JACC, FA9-B.

Ogunnaike, B. A. (1983): "A Statistical Appreciation of Dynamic Matrix Control", Proc. ACC, 1126.

Prett, D. M. and R. D. Gillette (1979): "Optimization and Constrained Multivariable Control of a Catalytic Cracking Unit", Paper 51b, AIChE 86th National Meeting, Los Angeles, California.

Richalet, J., A. Rault, J. L. Testud and J. Papon (1978): "Model Predictive Heuristic Control: Applications to Industrial Processes", Automatica, 14, 413.

Richalet, J. (1980): "General Principles of Scenario Predictive Control Techniques", Proc. JACC, FA9-B.

Reid, J. G., L. Gearhart and B. Schneble (1982): "Application of Output Predictive Algorithmic Control to Non-minimum Phase systems", Proc. CDC, 1026.

Rouhani, R. and R. K. Mehra (1982): "Model Algorithmic Control (MAC); Basic theoretical properties", Automatica, 18, 401.

4. Control of discrete MIMO systems with time-delays using the interactor matrix

4.1 Introduction

Time-delays are very common in process systems and there are numerous examples in the literature and textbooks illustrating the difficulties they cause, particularly in the design of controllers. The time-delays are often inherent in the process, the measurement transducers, the final control element and/or the discrete controller and usually cannot be easily removed or ignored.

In SISO systems where the time-delay is easily "factored out"; its effect is easy to understand, and its effect can be readily distinguished from the dynamic effects and gain effects. Design techniques, e.g. frequency domain, can easily accommodate time-delays and several special controllers, such as the Smith predictor (Smith (1957)) and Dahlin algorithm (Dahlin (1967)), have been developed to compensate for their effect on controllers.

In MIMO systems with time-delays the design of delay compensating control systems is more difficult. For example, if a step response is introduced into one or all of the inputs of a MIMO system then, in general, each output would exhibit a delay but it is not obvious how to predict or "factor out" the numeric value of the delays. The interactor factorization presented in this chapter provides a systematic *design* method for calculating the *minimum* delay that must be factored out for each output.

Many of the classical SISO design techniques for time-delay systems have been extended to the MIMO case. For example, Ogunnaike and Ray (1979) generalized the Smith predictor approach so that *all* the time-delays could be removed from characteristic equation (cf. continuous time case). Their examples showed that this approach gave better closed-loop performance than similar feedback control schemes without time-delay compensation. However, removing *all* the time-delays is *not* always the best approach. The interactor factorization used in this chapter results in better performance for the same examples used by Ogunnaike and Ray and can also be shown to be "optimal" for a significant class of problems.

In a recent paper by Jerome and Ray (1986), a new generalized multi-delay compensator (GMDC) was presented that improves the Ogunnaike and Ray (OR) compensator. Rather than trying to remove all the delays from the closed-loop characteristic equation as is done in the OR compensator, Jerome and Ray (1986) try to extend different interpretations of the SISO Smith predictor separately to the MIMO case. A delay-free characteristic equation is one consequence of the SISO Smith predictor which improves the closed-loop stability and which makes the controller design easier (especially for continuous time systems). However, there are other interpretations that can be extended to the MIMO case, such as: the SISO Smith predictor provides an immediate prediction of the effect of the control action on the process output to the controller; and it factors the process model into a delay term and a delay-free dynamic transfer function. These extensions lead to a factorization of the MIMO transfer function matrix, much like in the interactor factorization considered here. However, the "delay structure" defined by them is not general in scope and is based on the use of a pre-compensator and heuristic arguments. The advantages of the interactor factorization are that it is formal and that it applies in many other situations, not just feedback control with time-delay compensation. For example, the interactor factorization is also shown to assist with the interpretation, and provide a more systematic design method for applying techniques such as feedforward and one-step-ahead predictive control methods to MIMO time-delay systems. The interactor factorization can also be compared to the T_+T_- factorization proposed by Garcia and Morari (1983) in their Internal Model Control (IMC) but is again more general.

Note that the time-delay compensators suggested by Ogunnaike and Ray (1979) and Jerome and Ray (1986) can be applied to continuous as well as discrete time systems. The interactor factorization is defined only for discrete time systems and hence this method does not apply to continuous time systems. However, most time-delay compensators are implemented in discrete time using a digital computer eventually and therefore it is not a drawback in considering only the discrete time case.

The theory and properties of interactor factorization were presented by Wolovich and Falb (1976) and Wolovich and Elliott (1983) who showed that a rational, causal and full-rank system transfer function matrix $T(q^{-1})$ in the backward shift operator, q^{-1}, can always be factored such that $T(q^{-1})=\xi_T(q^{-1})R_T(q^{-1})$ where $\xi_T(q^{-1})$, called the inverse interactor matrix of $T(q^{-1})$, contains the basic delay structure of $T(q^{-1})$. This chapter does not extend the theoretical results of Wolovich and Falb (1976) and Wolovich and Elliott (1983) but rather shows how and why it can be used in the analysis and design of MIMO time-delay compensating control systems.

4.2 Properties of time-delay systems

The dynamical behaviour of discrete time systems is expressed in terms of difference equations, sometimes using forward shift notation or sometimes using backward shift notation. Alternatively, for conciseness of representation, the dynamical behaviour is often expressed as rational transfer functions (in the SISO case) or as rational transfer function matrices (in the MIMO case) either in the forward shift operator, q, or the backward shift operator, q^{-1}. These different representations are often interchangeable and equivalent, although involve some subtle consequences (see Wolovich and Elliott (1983) for a detailed exposition).

Causality is an important property in system description and its interpretation is slightly different in each representation. Control systems are required to be causal for realizability reasons and hence it is instructive to define this term rigorously before proceeding. In forward shift operator notation, a system is said to be causal if and only if the system transfer function or the system transfer function matrix is proper (i.e. finite at $q=\infty$). In backward shift operator notation, which is used in the rest of this chapter, a system is causal if and only if the system transfer function or the system transfer function matrix is finite at $q^{-1}=0$ (or at $q=\infty$).

SISO systems are discussed first because they provide a basis for illustrating and generalizing the concept for MIMO systems.

4.2.1 SISO systems

In SISO systems the concept of delay is straightforward and easy to understand. The delay can be easily separated from dynamic effects. Consider a discrete SISO system represented by a transfer function model as given below:

$$
\begin{aligned}
y(t) &= q^{-d}\theta(q^{-1})u(t) \\
&= q^{-d}\frac{B(q^{-1})}{A(q^{-1})}u(t) \\
&= q^{-d}\frac{b_1 + \ldots + b_n q^{-n+1}}{1+a_1q^{-1} + \ldots + a_nq^{-n}}u(t) \qquad (1)
\end{aligned}
$$

where $\theta(q^{-1})$ is a rational transfer function or an element of a rational transfer function matrix $T(q^{-1})$. Here $d=k+1$ represents the system delay in number of sampling intervals, where k is the pure time-delay of the physical system and the extra unit delay is due to discrete sampling and zero order hold. From this relation one can observe the following facts:

1) $\lim_{q\to\infty} \theta(q^{-1}) = \lim_{q\to\infty} \frac{B(q^{-1})}{A(q^{-1})} = b_1 \neq 0$

   In other words b_1 can be interpreted as the high frequency gain of the system;

2) $y(\infty) = \theta(1)\, u(\infty)$, where $\theta(1) = \frac{b_1 + \ldots + b_n}{1+a_1 + \ldots + a_n}$ is the system steady state gain;

3) $\theta(q^{-1})$ is bicausal (i.e. it has neither poles nor zeros

at infinity, which means it can be inverted without any predictive terms);

4) the system has d zeros at $q=\infty$.

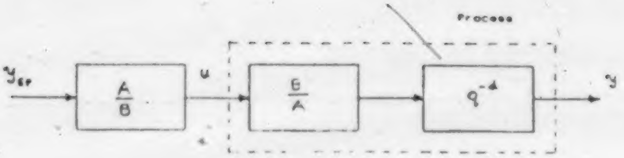
Thus the delays in a discrete SISO system are characterized by zeros at infinity. When the delay is removed, the system model, $\theta(q^{-1})$, becomes causally invertible (i.e. $\theta^{-1}(q^{-1})$ is finite at $q=\infty$). Perfect setpoint tracking can therefore be obtained, in an open-loop or feedforward configuration using the inverse of $\theta(q^{-1})$ as the controller as shown in Figure 4.1(a). The output follows the setpoint exactly but with a delay of d sampling intervals.

4.2.2 MIMO systems

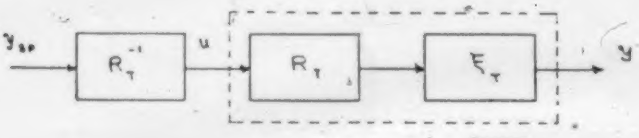
Using the SISO discussion as a basis it is easy to understand or generalize the concept of delay to MIMO systems. However, in the MIMO case, the delay is characterized by a polynomial matrix. Suppose that a system with l inputs and p outputs is represented by a transfer function matrix which has the form:

$$T(q^{-1}) := \begin{bmatrix} q^{-d_{11}}\theta_{11}(q^{-1}) & \ldots & q^{-d_{1l}}\theta_{1l}(q^{-1}) \\ \cdot & & \\ \cdot & & \\ \cdot & & \\ q^{-d_{p1}}\theta_{p1}(q^{-1}) & \ldots & q^{-d_{pl}}\theta_{pl}(q^{-1}) \end{bmatrix} \tag{2}$$

Note that $d_{ij} = k_{ij}+1$ where k_{ij} is due to the pure time-delay between i^{th}-output and j^{th}-input. $T(q^{-1})$ is assumed to be causal and full-rank (or when $T(q^{-1})$ is square, i.e. $l=p$, $T(q^{-1})^{-1}$ exists although it may not be causal), implying


Process
y_{sp}
$\frac{A}{B}$
u
$\frac{B}{A}$
q^{-d}
y

(a) SISO case


y_{sp}
R_T^{-1}
u
R_T
ξ_T
y
Process

(b) MIMO case

Figure 4.1 Open-loop inverse model control for perfect setpoint following

functional controllability. This means that all the system outputs can be made to track their respective reference trajectories.

One can easily see from the above SISO discussion that the $(i,j)^{th}$ pair has a delay of d_{ij} sampling intervals and that $\theta_{ij}(q^{-1})$ is causally invertible. However it is not immediately obvious how to factor this MIMO relation into a "delay structure" multiplied by a dynamic relation as in the SISO case.

Wolovich and Falb (1976) showed that for every pxl, proper, rational transfer function matrix, $T'(q)$ in the forward shift operator q, there exists a unique, pxp, lower left-triangular polynomial matrix, $\xi'_{T'}(q)$, called the interactor matrix of $T'(q)$ such that $\det\{\xi'_{T'}(q)\}=q^d$ and

$$\lim_{q\to\infty} \xi'_{T'}(q)\ T'(q) = K_{T'}$$

which is a finite, full-rank (=min{l,p}), real matrix. Here d is the number of infinite zeros in the multivariable system.

As in Wolovich and Elliott (1983) a similar observation can be made regarding a causal, full-rank, rational transfer function matrix, $T(q^{-1})$, in the backward shift operator q^{-1}. Associated with every $T(q^{-1})$ there exists a factorization of the form

$$T(q^{-1}) = \xi_T(q^{-1})\ R_T(q^{-1}) \tag{3}$$

where

1) $\xi_T(q^{-1})$ is a pxp, polynomial matrix that exists, is unique and is non-singular and which is called the inverse interactor matrix of $T(q^{-1})$ (to distinguish the fact that it is associated with q^{-1} notation as opposed to q notation);

2) $\lim_{q\to\infty} \xi_T(q^{-1})^{-1}\, T(q^{-1}) = \lim_{q\to\infty} R_T(q^{-1}) = K_T$ is a finite, full-rank constant matrix (K_T represents the high frequency gain matrix of the system);

3) $\det\{\xi_T(q^{-1})\} = q^{-d}$, where d is the number of infinite zeros in the system;

4) $\xi_T(q^{-1})^{-1} = \xi'_{T'}\overline{(q)}$ and $\xi_T(q^{-1})^{-1}$ is a stable operator;

5) $R_T(q^{-1})$ is bicausal which means that it has only finite poles and zeros and which as in the SISO case means that $R_T(q^{-1})$ is causally invertible (when the number of inputs is equal to the number of outputs or $R_T(q^{-1})$ is square).

The inverse interactor, $\xi_T(q^{-1})$, can be defined and determined directly from $T(q^{-1})$ and Wolovich and Falb (1976) provide a method for performing the factorization for any arbitrary rational transafer function matrix in q. This procedure is summarized in an appendix to this chapter and can be applied to arbitrary rational transfer function matrices in q^{-1} with little difficulty.

The inverse interactor, $\xi_T(q^{-1})$, defines a left divisor of the transfer function matrix, $T(q^{-1})$, factoring out all the infinite zeros (or the time-delays) of the MIMO system. From the previous discussion on SISO systems it is immediately obvious that the inverse interactor characterizes the "delay structure" of a discrete MIMO system. $R_T(q^{-1})$ is referred to as a "residual". To satisfy fact (2) in the SISO discussion one can redefine the residual as:

$$\hat{R}_T(q^{-1}) := \xi_T(1)\ R_T(q^{-1}) \tag{4}$$

so that one has

6) $y(\infty) = T(1)\ u(\infty) = \hat{R}_T(1)\ u(\infty)$

As in the SISO case perfect setpoint tracking can be achieved in an open-loop with $R_T(q^{-1})^{-1}$ as a controller as shown in Figure 4.1(b). In this case, $y=\xi_T(q^{-1})y_{sp}$ which means that the system output tracks the setpoint with delays present in $\xi_T(q^{-1})$.

The inverse interactor may be a left or right factor of the system transfer function matrix and may be lower or upper triangular in the most general case with polynomial elements. But for the chosen factorization and structure it is unique. Since a left factor represents the delays as occurring at the output which naturally leads to predictive control, it is used throughout in the rest of the chapter. It is given by:

$$\xi_T(q^{-1}) := H_T(q^{-1})\ \mathrm{diag}(q^{-d_1}, \ldots, q^{-d_p}) \tag{5}$$

where

$$H_T(q^{-1}) := \begin{bmatrix} 1 & 0 & \ldots & 0 \\ h_{21}(q^{-1}) & 1 & \ldots & 0 \\ \cdot & \cdot & & \\ \cdot & & & \\ h_{p1}(q^{-1}) & h_{p2}(q^{-1}) & \ldots & 1 \end{bmatrix}$$

with H_T unimodular, $d_i \geq 1$ $\forall i$ and $\{h_{ij}(q^{-1})\}$ are polynomials with real coefficients (note that $\Sigma d_i =: d$ is the number of infinite zeros in the system). Often $H_T(q^{-1})=I$ and when this is the case then each d_i represents the natural delay between the i^{th}-output and the system input (Wolovich and Elliott (1983)).

The above interpretation of the inverse interactor as representing the "delay structure" in MIMO systems is the basis for the use of the interactor factorization given by equation (3) in the following sections. In particular its role in feedback control with time-delay compensation, feedforward control, model following and multivariable predictive control are considered.

4.3 Time delay compensation in feedback systems

It is well known that satisfactory control of time-delay systems is very difficult to achieve but can often be improved by using some form of delay compensation. The Smith predictor (SP) proposed by Smith (1957) was one of the earliest time-delay compensators proposed for SISO delayed systems. Employing a process model, Smith showed that the closed-loop characteristic equation in the continuous time case can be made delay free which allows higher gains to be used and also which makes the controller easier to design. Essentially the Smith predictor nullifies the effect of the time-delays on the closed-loop system by providing phase advance through prediction using the delay-free process model. Alevisakis and Seborg (1973) extended Smith's idea to multivariable systems having a single common delay (i.e. the systems with the inverse interactor given by $\xi_T(q^{-1})=q^{-d}I$, where d is the delay in each element). Ogunnaike and Ray (1979) then generalized the SP technique to multivariable systems having multiple delays by removing all delays from the closed-loop characteristic equation using essentially the transfer function matrix of the plant with the delays in all (1,1) minors removed as a predictor just as in the SISO SP. Although this scheme results in better closed-loop performance as compared to the performance without any delay compensation, in general it does not lead to an optimal design.

4.3.1 General formulation

A general SP scheme is shown in Figure 4.2. It applies to SISO as well as MIMO systems which may be either continuous or sampled. The "delay factored" plant model, i.e. the plant model with the appropriate "delay structure" factored out, is referred to as the "predictor" and designated as P. The controller T_C is normally designed for the feedback control of the delay factored plant model, P, (not the plant T or model T_m). In the case of perfect modeling where $T_m=T$ then $y_e=v$ and the feedback loop effectively involves only T_C and P.

When P approximates the actual plant T except for delays in the plant output variables then the actual plant outputs y(t) will simply be delayed versions of the predictor outputs $y_P(t)$ and the "best" control of P will give the "best" control of T. More generally the relation between the actual plant outputs y and the predictor outputs y_P can be used to analyze the predictor scheme represented by Figure 4.2. The following expressions can be obtained

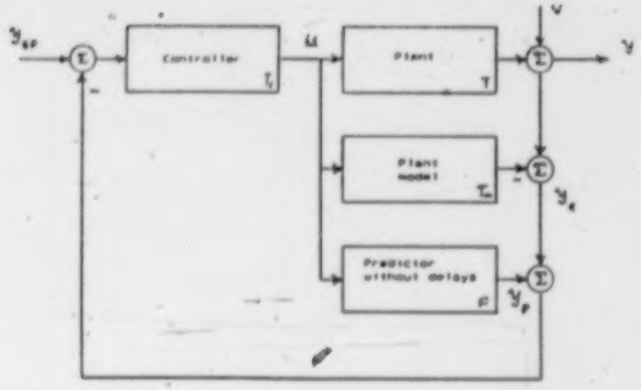

y_{sp}
Σ
$-$
Controller
T_l
u
Plant
T
v
Σ
y
Plant
model
T_m
$-$
Σ
y_e
Predictor
without delays
F
y_p
Σ

Figure 4.2 Structure of the generalized Smith predictor

from Figure 4.2.

$$y = T\,u + v \tag{6}$$

$$y_P = P\,u \tag{7}$$

$$u = (I+T_CP)^{-1}(T_Cy_{sp}-T_Cv) \tag{8}$$

$$y_P = P(I+T_CP)^{-1}(T_Cy_{sp}-T_Cv) \tag{9}$$

$$y = T(I+T_CP)^{-1}T_Cy_{sp} + [I-T(I+T_CP)^{-1}T_C]v \tag{10}$$

For the case of no disturbance, i.e. v=0,

$$y = TP^{-1}\;y_P \tag{11}$$

4.3.2 SISO Smith predictor

In the original scheme of Smith, for continuous SISO systems, the plant $\theta(s)$, is given by:

$$\theta(s) = e^{-ds}\;\theta_1(s) \tag{12}$$

where $\theta_1(s)$ is a rational, scalar transfer function and d is the delay. The plant model assuming perfect modeling is $\theta_m(s) = \theta(s)$, and the predictor $P = \theta_1(s)$ (delay-factored or delay-free). The controller $\theta_C(s)$ is typically a PI(D) controller. Equation (11) can then be written as:

$$y(s) = (\theta_1(s)e^{-ds})\theta_1(s)^{-1}y_P(s)$$

$$= e^{-ds}\ y_p(s) \tag{13}$$

Thus when the disturbance $v(s)=0$, then $y(s)=e^{-ds}y_p(s)$ and *the performance of the plant is the same as that of* y_p *except for the delay* d.

4.3.3 MIMO time-delay compensators

Extending the SP techniqe to the MIMO case requires an interpretation of what characterizes the delay structure of the system and what is an appropriate predictor. Several MIMO SP techniques have been proposed. In the scheme of Ogunnaike and Ray (1979) (from now on referred to as OR compensator for brevity) the plant is a MIMO continuous or sampled data system with multiple delays. It is represented by a pxl, rational transfer function matrix $T(\cdot)$ where $\cdot$ represents s (continuous case) or q^{-1} (sampled data case). The plant model is $T_m=T$ (assuming perfect modeling). The predictor in this case is the plant model with the delays of all the (1,1) minors removed and is denoted by T_m^* (i.e. $P=T_m^*=T^*$). The controller T_c is typically a classical feedback controller designed for the model P (or T^*) just as for the SISO SP. When there is no disturbance, then equation (11) gives for this case:

$$y = T\ T^{*^{-1}}\ y_p \tag{14}$$

In general the product $TT^{*^{-1}}$ *has no special form and* y *is* *not* *just a delayed version of* y_p *and hence this is* *not* *a true MIMO extension of the SP*. Note that in the continuous case the closed-loop characteristic polynomial (i.e. numerator of $\det\{I+T_C T^*\}$) does not contain any delay terms. This makes the design of the controller T_C based on y_P easier as well as improves the stability of the closed-loop (i.e. allows higher gains to be used in T_C), but y_P is *not* the actual variable of interest.

An improvement of this simple multi-delay compensator is the generalized multi-delay compensator (GMDC) proposed by Jerome and Ray (1986) which again applies to both continuous as well as discrete MIMO processes. Their design is based on extending different *properties* or *interpretations* of the SP for the SISO case to the MIMO case using heuristic arguments. The exact form which the GMDC assumes depends on which of the properties of the SISO SP scheme is being extended to the MIMO case. These properties may be summarized for discrete systems as follows:

Property 1: SISO SP eliminates the time-delay from the closed-loop characteristic equation. This gives the OR compensator which is a special case of GMDC.

Property 2: For setpoint changes the SISO SP provides an immediate prediction of the effects of its control action on the system output forecast d time intervals into the future, where d is the process time-delay. In the MIMO case this property leads to choosing $y_{P_i}=y_i(t+d_i)$ for the i^{th}-output where $d_i :=$ minimum (with respect to j) of $\{d_{ij}\}$. Alternatively stated, this implies the use of a diagonal "delay structure", $\mathrm{diag}(q^{-d_1}, \ldots, q^{-d_p})$.

Property 3: The SISO SP factors the plant model into two parts: a non-invertible part (i.e. a delay) and a part which is invertible without predictive terms. To extend this concept to MIMO systems a precompensator, $D(q^{-1})$, is added to the plant, $T(q^{-1})$; and $T_F(q^{-1})$ is factored out from $T_m(q^{-1})D(q^{-1})$ to be used as the predictor such that both $T_F(q^{-1})$ and $T_F(q^{-1})^{-1}$ contain no predictive terms. Note that $T_F(q^{-1})$ is a right divisor of $T_m(q^{-1})D(q^{-1})$. The precompensator, $D(q^{-1})$, may add additional delays to the open-loop plant. Also note that this does not automatically factor the original plant into a "delay structure" multiplied by a dynamic relationship. They outline a heuristic procedure for selecting $T_F(q^{-1})$ and $D(q^{-1})$.

4.3.4 Interactor factorization

For *discrete* MIMO processes with (multiple) delays the interactor factorization provides an alternative method for designing time-delay compensators, essentially achieving properties (2) and (3) above. The advantage of using the interactor factorization is that it is based on more formal arguments and applies in many other situations as well as making the various interpretations simpler. Suppose that the plant is given by (cf. equation (3))

$$T(q^{-1}) = \xi_T(q^{-1})R_T(q^{-1}) \qquad (15)$$

where $\xi_T(q^{-1})$ is the inverse interactor of $T(q^{-1})$. If perfect modeling is assumed then the plant model is:

$$T_m(q^{-1}) = T(q^{-1}) \qquad (16)$$

Then a natural choice of the predictor shown in Figure 4.2 is

$$P(q^{-1}) = \xi_{T_m}(1)R_{T_m}(q^{-1}) = \hat{R}_{T_m}(q^{-1}) = \hat{R}_T(q^{-1}) \tag{17}$$

where $\xi_{T_m}(1)$ ensures that the steady state gains of $P(q^{-1})$ and $T_m(q^{-1})$ are the same (i.e. $P(1)=T_m(1)=T(1)$). The controller $T_C(q^{-1})$ can be a classical PID type controller or a inverse model (predictive) controller based on $P(q^{-1})$. From equation (11) for the case of no disturbance it follows that

$$\begin{aligned} y(t) &= \xi_T(q^{-1})R_T\hat{R}_T(q^{-1})^{-1}y_P(t) \\ &= \xi_T(q^{-1})\xi_T(1)^{-1}y_P(t) \end{aligned} \tag{18}$$

If $\xi_T(q^{-1})$ is diagonal (which is quite common) then

$$y(t) = \mathrm{diag}(q^{-d_1}, \ldots, q^{-d_P})\ y_P(t) \tag{19}$$

The performance of the plant is the same as y_P except for the time-delays and hence *this approach can be interpreted as a MIMO extension of the SISO Smith predictor (cf. equation (13)). If the controller $T_C(q^{-1})$ is designed such that the performance of $y_P(t)$ is optimal, then $y(t)$ will also be optimal in the same sense since it is identical except for the delays which can be shown to be minimal.* However, the design of the controller is based on $P(q^{-1})$ which in the general case may contain delays in some elements. This may complicate the design of $T_C(q^{-1})$ but does not change the basic idea. The OR compensator and the interactor factorization results in the same Smith predictor only when each row of $T(q^{-1})$ has a common delay and $\xi_T(q^{-1})$ is diagonal. When $\xi_T(q^{-1})$ is diagonal the Smith predictor based on interactor factorization *satisfies property (2) of Jerome and Ray's GMDC.*

Thus the interactor factorization leads to a logical extension of the SISO SP scheme to MIMO systems. Some simulated examples are presented later to illustrate some of the points made here.

For discrete systems the controller $T_C(q^{-1})$ can be designed in several ways. If the controller is diagonal (i.e. multi-loop) then pairing may be important. *The interactor factorization provides natural pairing in that case because there is at least one input for each output having a delay free transmittance (cf. fact (2) in section 4.2.2).* In general the question of pairing applies to many methods and should not be the basis for choosing between them.

Since y_p depends on the closed-loop performance of the system involving $P(q^{-1})$ and $T_C(q^{-1})$ in Figure 4.2, it is not obvious which controller will yield the best results. Suppose $T_C(q^{-1})$ is a inverse model type of (or ∞-gain, i.e. $\|T_C\|\to\infty$) controller (cf. Figure 4.3) so that the output of the predictor reaches the setpoint immediately in a dead-beat manner. Then

$$y_p(t) = y_{ref}(t) \tag{20}$$

and where $y_p(t)$ is the plant output before the inverse

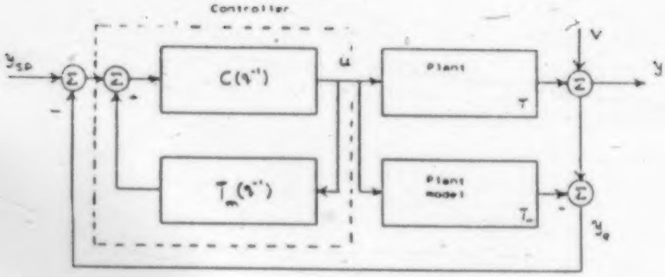

Controller
y_{sp}
Σ
-
Σ
-
$C(q^{-1})$
u
$T_m(q^{-1})$
Plant
T
Plant model
T_m
v
Σ
y
Σ
-
y_e

Figure 4.3 Internal model controller (IMC)

interactor as shown in Figure 4.1.

$$y(t) = T(q^{-1})u(t) = \xi_T(q^{-1})R_T(q^{-1})[\xi_T(1)R_T(q^{-1})]^{-1}y_{ref}(t)$$

$$= \xi_T(q^{-1})\xi_T(1)^{-1}y_{ref}(t) \qquad (21)$$

If the inverse interactor matrix is diagonal,

$$y(t) = \mathrm{diag}(q^{-d_1}, \ldots, q^{-d_p})y_{ref}(t) \qquad (22)$$

or

$$y_i(t) = y_{ref_i}(t-d_i), \; i=1, \ldots, p \qquad (23)$$

That is, each output of the process reaches its corresponding reference input exactly after a number of sampling intervals equal to its minimal time-delay as defined by the inverse interactor matrix. This is the best possible for discrete systems. A similar relation can be obtained for the OR compensator (for the dead-beat control case) which is given by

$$y(t) = T(q^{-1})T^*(q^{-1})^{-1} \, y_{ref}(t) \qquad (24)$$

From equation (24) clearly the OR compensator does not guarantee optimality in the sense that each plant output does not track the corresponding setpoint with only a minimal delay as in the interactor factorization scheme. Note also that Smith predictor scheme with a diagonal inverse interactor matrix is equivalent to that of Jerome and Ray (1986) satisfying property 2.

So far the discussion has been limited to the diagonal interactor case, which results in optimal time-delay compensation. In the general case, the interactor is lower left triangular and the relation between y and y_P can be expressed (for $v=0$) as:

$$y(t) = H_T(q^{-1}) \, \mathrm{diag}(q^{-d_1}, \ldots, q^{-d_P}) \, H_T(1)^{-1} \, y_P(t) \qquad (25)$$

Except for the first output, the others are interacting and a general result regarding optimality is difficult to deduce. However, all the delays in the inverse interactor ξ_T are minimal, and if y_P is optimal in some sense then y will follow y_P with minimum delays but with perhaps some interaction. By a *reordering* of the output vector, y, before carrying out the factorization, any single desired output can be made non-interacting.

4.3.4.1 Decoupled response when the inverse interactor is triangular

The class of systems for which the inverse interactor is diagonal and leads to a closed-loop decoupled system, e.g. when a high gain or inverse model controller is used, is usually large. However, systems with triangular inverse interactor are not uncommon. If a decoupled response is desired when the interactor is triangular, then one approach is the use of a pre- and/or post-compensator as illustrated by Figure 4.4. Such a compensator adds extra delays to the

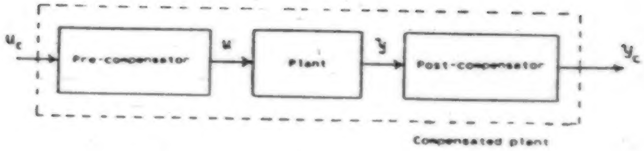

u_c
Pre-compensator
u
Plant
y
Post-compensator
y_c
Compensated plant

Figure 4.4 Use of pre- and/or post-compensators for achieving decoupling

plant at the input or at the output as the case maybe so that the compensated plant model has a diagonal interactor. Singh and Narendra (1984) present a pre-compensator design. Shah *et al*. (1986) consider the design of both pre- and post-compensators. *Note that adding a pre-compensator to diagonalize the inverse interactor makes the interactor factorization approach equivalent to the GMDC of Jerome and Ray (cf. Property 3).*

In general, a compensator may introduce arbitrarily large delays into the closed-loop system and may not be desirable. In such a case it may be better to tolerate some interactions as defined by equation (25). Whether to use a decoupling compensator or not is best decided after some simulations. *The GMDC design approach does not provide this choice.*

4.3.4.2 Comparison with T_+T_- factorization of IMC

Garcia and Morari (1983) showed that the Smith predictor is a subset of a general model based control scheme which they called IMC (Internal Model Control). In their analysis the Smith predictor (Figure 4.2) is rearranged as shown in Figure 4.3 with the controller $T_c(q^{-1})$ chosen as the inverse of the plant model, to achieve perfect control for setpoint changes. However a causal controller $T_c(q^{-1})$ equal to the exact inverse model cannot be implemented if the plant model $T_m(q^{-1})$ contains time-delays or unstable zeros (zeros outside the unit circle). They therefore proposed a factorization of the form $T_m(q^{-1})=T_+(q^{-1})T_-(q^{-1})$ where $T_-(q^{-1})^{-1}$ is causal, with stable and well damped poles. The IMC scheme (Figure 4.3) is characterized by

Plant $T(q^{-1})$ (includes delays)

Model

$T_m(q^{-1})=T(q^{-1})$ (perfect modeling)
$=T_+(q^{-1})T_-(q^{-1})$ (26)

Controller $T_C(q^{-1})=T_-(q^{-1})^{-1}$ (27)

In general $T_+(q^{-1})$ factors out the time-delays plus the unstable and poorly damped zeros of $T_m(q^{-1})$. Since the factorization of the delays and other finite zeros can be done independently, assume for simplicity, $T_+(q^{-1})$ factors out only delays. The inverse interactor $\xi_{T_m}(q^{-1})$ can be related to $T_+(q^{-1})$ whose design is based on achieving decoupling and extracting minimum delays. For the case when $\xi_{T_m}(q^{-1})$ is diagonal, both factorizations are identical and lead to optimal design. However in general $T_+(q^{-1})$ factors out larger delays than necessary (i.e. closed-loop system will have larger delays than necessary) when the inverse interactor is triangular. On the other hand $\xi_{T_m}(q^{-1})$ extracts only minimum delays required but may result in some interactions between the j^{th}-setpoint and the i^{th}-output ($i \neq j$). The fact that T_+ decouples the closed-loop may be an advantage but it can introduce arbitrarily large extra delays into the loop, which is undesirable.

To illustrate, consider Example 1 in Table 4.1. The IMC design (T_+) results in the following closed-loop relation

$$y(t)=T_+(q^{-1})y_{ref}(t)= \mathrm{diag}(q^{-5}, q^{-6})y_{ref}(t) \qquad .(28)$$

The inverse interactor for this example is

$$\xi_{T_m}(q^{-1}) := \begin{bmatrix} q^{-3} & 0 \\ 0.5q^{-4} & q^{-6} \end{bmatrix} \qquad (29)$$

and the interactor factorization leads to the closed-loop relation

$$\begin{aligned} y(t) &= \xi_{T_m}(q^{-1})\xi_{T_m}(1)^{-1}y_{ref}(t) \\ &= \begin{bmatrix} q^{-3} & 0 \\ .5q^{-4}(1-q^{-2}) & q^{-6} \end{bmatrix} y_{ref}(t) \end{aligned} \qquad (30)$$

The IMC design introduced 2 sample periods of extra delay in output 1 to achieve decoupling. The interactor design has minimal delays in both outputs but shows some interaction in the 2nd output.

Note that the T_+T_- factorization is a specific design tool for IMC. It is not a general factorization scheme, e.g. for use in a Smith predictor, because $T_-(q^{-1})$ in general *may not be causal*. Also the IMC

Table 4.1 A list of the models used in the illustrative examples with interactor factorization

Example	Process model		Interactor Factorization	
	$T_1(q^{-1})$	$T_m(q^{-1})$	$\xi_{T_m}(q^{-1})$	$R_{T_m}(q^{-1})$
1.	--	$\frac{1}{\Delta_1}\begin{bmatrix} q^{-5} & q^{-3} \\ q^{-6} & .5q^{-4} \end{bmatrix}$	$\begin{bmatrix} q^{-3} & 0 \\ .5q^{-4} & q^{-6} \end{bmatrix}$	$\frac{1}{\Delta_1}\begin{bmatrix} q^{-2} & .1 \\ .5 & 0 \end{bmatrix}$
2.	$\frac{1}{\Delta_2}\begin{bmatrix} 3q^{-4} & 2q^{-5} \\ -q^{-4} & -4q^{-5} \end{bmatrix}$	$\frac{1}{\Delta_2}\begin{bmatrix} 2q^{-3} & q^{-4} \\ 1.5q^{-4} & -3q^{-5} \end{bmatrix}$	$\begin{bmatrix} q^{-3} & 0 \\ \frac{3}{4}q^{-4} & q^{-5} \end{bmatrix}$	$\frac{1}{\Delta_2}\begin{bmatrix} 2 & q^{-1} \\ 0 & -3.75 \end{bmatrix}$
3.	$\frac{1}{\Delta_3}\begin{bmatrix} 3q^{-1} & 2q^{-1} \\ -q^{-1} & -4q^{-1} \end{bmatrix}$	$\frac{1}{\Delta_3}\begin{bmatrix} 2q^{-3} & q^{-4} \\ 1.5q^{-4} & -3q^{-5} \end{bmatrix}$	$\begin{bmatrix} q^{-3} & 0 \\ \frac{3}{4}q^{-4} & q^{-5} \end{bmatrix}$	$\frac{1}{\Delta_3}\begin{bmatrix} 2 & q^{-1} \\ 0 & -3.75 \end{bmatrix}$
4. Distillation Column	$\begin{bmatrix} \frac{.255q^{-8}}{1-.933q^{-1}} \\ \frac{.3712q^{-6}}{1-.927q^{-1}} \end{bmatrix}$	$\begin{bmatrix} \frac{.8975q^{-2}}{1-.9419q^{-1}} & \frac{-.8719q^{-4}}{1-.9535q^{-1}} \\ \frac{.5786q^{-8}}{1-.912q^{-1}} & \frac{-1.301q^{-4}}{1-.933q^{-1}} \end{bmatrix}$	$\begin{bmatrix} q^{-2} & 0 \\ 0 & q^{-4} \end{bmatrix}$	$\begin{bmatrix} \frac{.8975}{1-.9419q^{-1}} & \frac{-.8719q^{-2}}{1-.9535q^{-1}} \\ \frac{.5786q^{-4}}{1-.912q^{-1}} & \frac{-1.301}{1-.933q^{-1}} \end{bmatrix}$
5. Chemical Reactor	$\frac{1}{\Delta_4}\begin{bmatrix} .1063q^{-1} & -.0244q^{-2} \\ .032q^{-1} & +.012q^{-2} \end{bmatrix}$	$\frac{1}{\Delta_4}\begin{bmatrix} .2658q^{-4} & -.061q^{-5} & .012q^{-5} & +.044q^{-6} \\ .085q^{-5} & +.03q^{-6} & .1063q^{-6} & -.0244q^{-7} \end{bmatrix}$	$\begin{bmatrix} q^{-4} & 0 \\ .307q^{-5} & q^{-6} \end{bmatrix}$	See Table 5.2

$\Delta_1 := 1 + .5q^{-1}$

$\Delta_2 := 1 - .8q^{-1}$

$\Delta_3 := 1 - .8q^{-1}$

$\Delta_4 := 1 - .5q^{-1} + .0498q^{-2}$

design method for achieving decoupling is not equivalent to the pre-compensator method suggested by Singh and Narendra (1982).

4.3.5 A binary distillation column example

Distillation columns have been recognized as very complex systems to control satisfactorily because of strong interactions and the presence of dead times. The present example is a methanol-water pilot scale binary distillation column, which is characterized by large multiple delays (Wood and Berry (1973)). It has been considered extensively in the literature, particularly in connection with delay compensation studies, including Ogunnaike and Ray (1979).

The two outputs to be controlled are top and bottom product compositions, with reflux and steam flow rates as the manipulated inputs (variables y_1, y_2, u_1 and u_2 respectively). The feed flow rate and composition are the disturbances in the column (variables v_1 and v_2 respectively). All the variables are represented as deviations from their corresponding steady state values, which can be found in Wood and Berry (1973). In the same reference, a dynamic model of the column as a continuous transfer function matrix is given. A discrete version with a sampling time $T_0=1$ minute is given Table 4.1 (Example 4). For this example, the inverse interactor $\xi_{T_m}(q^{-1})=\mathrm{diag}(q^{-2},q^{-4})$. In Table 4.2 the predictors used in the OR and IF schemes are presented (since the inverse interactor is diagonal the Smith predictors of Jerome and Ray (1986) and IF are equivalent).

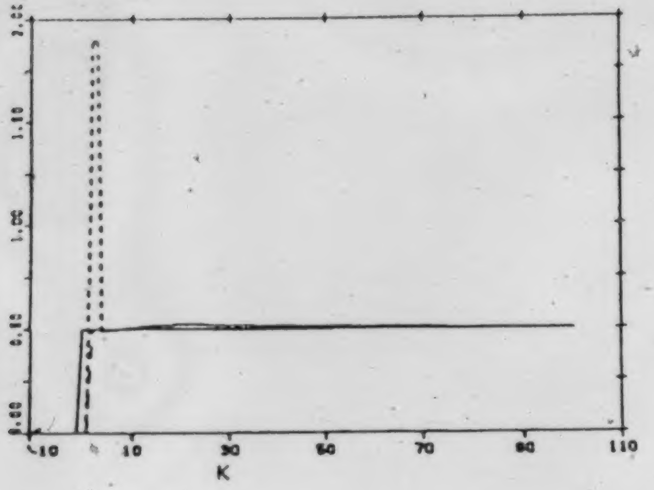
Table 4.2 The predictors used in OR and IF design approaches of the SP scheme for the distillation column and reactor examples

Example	$P(q^{-1})$	
	Interactor Factorization Approach	OR Approach
Distillation Column	$\begin{bmatrix} \frac{.6975q^{-1}}{1-.9419q^{-1}} & \frac{-.879q^{-2}}{1-.9533q^{-1}} \\ \frac{.5786q^{-4}}{1-.9123q^{-1}} & \frac{-1.301q^{-1}}{1-.933q^{-1}} \end{bmatrix}$	$\begin{bmatrix} \frac{.6975q^{-1}}{1-.9419q^{-1}} & \frac{-.879q^{-1}}{1-.9535q^{-1}} \\ \frac{.5786q^{-1}}{1-.9123q^{-1}} & \frac{-1.301q^{-1}}{1-.933q^{-1}} \end{bmatrix}$
Chemical Reactor	$\begin{bmatrix} \frac{.2568q^{-1}-.061q^{-2}}{\Delta} & \frac{.012q^{-2}+.004q^{-3}}{\Delta} \\ \frac{.1284q^{-1}-.01836q^{-2}}{\Delta} & \frac{.1027q^{-1}-.0221q^{-2}+.0013q^{-3}}{\Delta} \end{bmatrix}$	$\begin{bmatrix} \frac{.2658q^{-1}-.061q^{-2}}{\Delta} & \frac{.012q^{-1}+.004q^{-2}}{\Delta} \\ \frac{.08q^{-1}+.03q^{-2}}{\Delta} & \frac{.1063q^{-1}-.024q^{-2}}{\Delta} \end{bmatrix}$

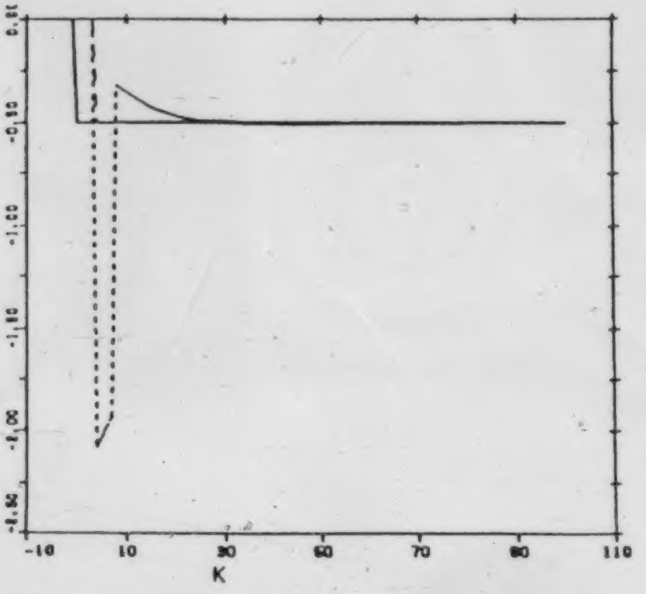
$\Delta := 1 - .5q^{-1} + .0498q^{-1}$

When the controller is an inverse model controller (e.g. IMC), then the structure in Figure 4.3 is used with $T_c(q^{-1})=P(q^{-1})^{-1}$. Since the inverse interactor for this example is diagonal, optimal performance of the actual plant should be obtained with the IF scheme. Figure 4.5 shows the simulated column responses for setpoint changes of 0.5 and -0.5 in top and bottom compositions respectively, for both the schemes. Note that the discrete transfer function model of the distillation is used in the simulations. The IF approach results in dead-beat response after minimum delays of 2 and 4 sample intervals in y_1 and y_2 respectively. As compared with this, the OR scheme exhibits large overshoots and longer settling times.

The Smith predictor (Figure 4.2) was also evaluated with the controller $T_c(q^{-1})$ as a diagonal PI controller (i.e. $T_c(q^{-1})=\text{diag}((q_0{}^i+q_1{}^iq^{-1})/(1-q^{-1}))$. The controller parameters were tuned based on the performance of the predictor output y_p. The controller parameters presented in Table 4.3 were used which resulted in approximately similar servo performance for y_p in both OR and IF schemes. (Slightly higher gains were used with OR scheme deliberately.) With these controller parameters, the closed-loop response of the simulated plant are shown in Figures 4.6 to 4.9, for both setpoint and disturbance changes. For setpoint changes, the IF approach gives simulated plant output responses which are identical to y_p (except for delays) and is quite satisfactory. The responses of the simulated plant with OR delay compensator are characterized by considerable overshoot for setpoint changes (because of the interactions in TT^{*-1}). The IF approach again shows better performance



(a) Output 1 (i.e. y_1)

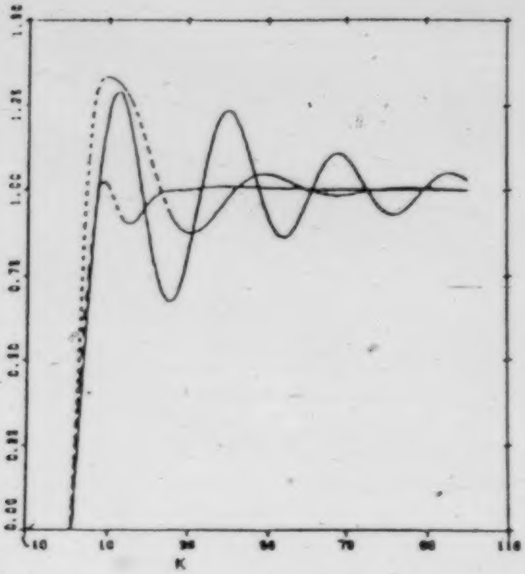


(b) Output 2 (i.e. y_2)

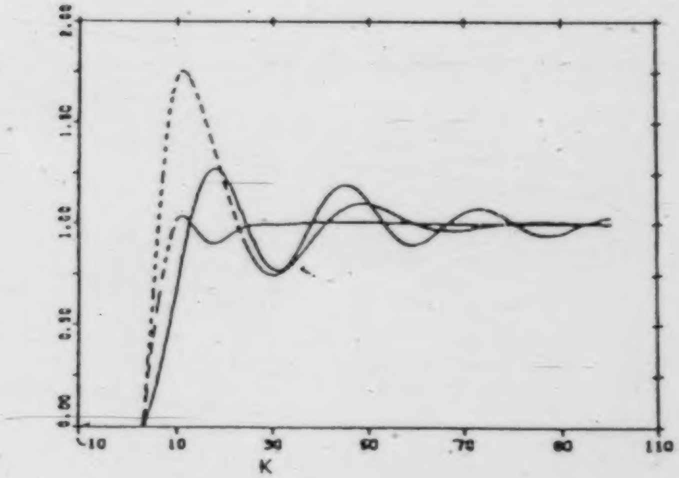
Figure 4.5 Simulated responses of the distillation column for dead-beat control and for setpoint changes; OR: dashes, IF: continuous

Table 4.3 PI controller parameters for the distillation column example

	Loop 1		Loop 2	
	q_0	q_1	q_0	q_1
OR approach	0.45	-0.3	-0.25	0.15
IF approach	0.35	-0.3	-0.18	0.15
No delay compensation	0.2225	-0.1775	-0.0475	0.0325

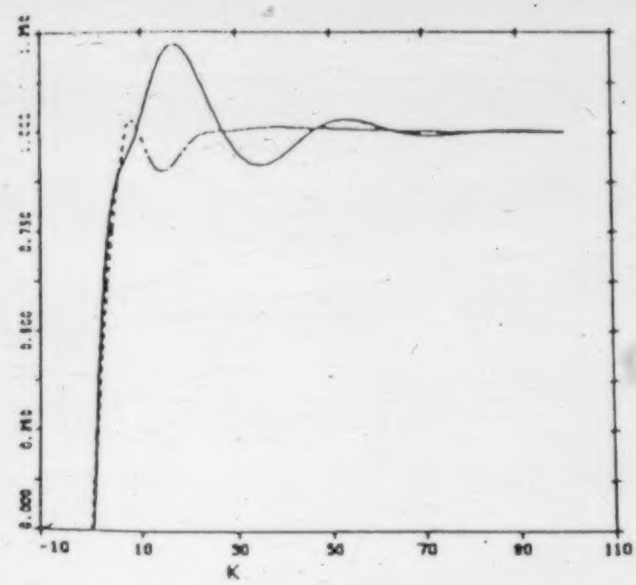


(a) $y_{set}=(1\ 0)^T$

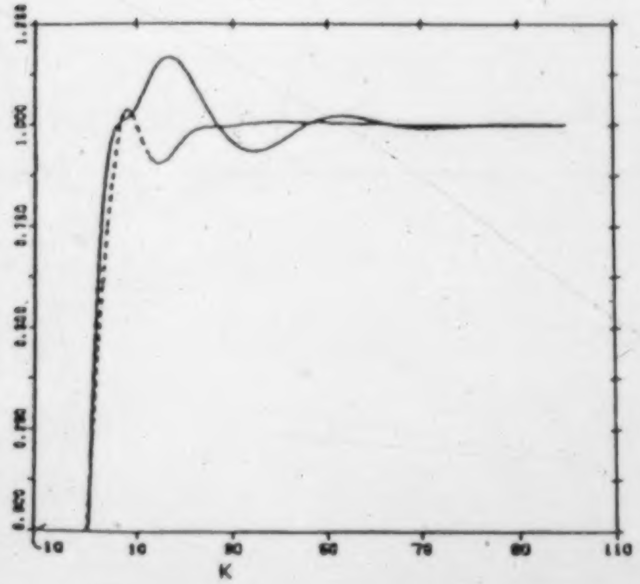


(b) $y_{set}=(0\ 1)^T$

Figure 4.6 Simulated responses of the distillation column for delay compensated multi-loop PI control and for setpoint changes; no compensation: continuous, OR: small dashes, IF: large dashes

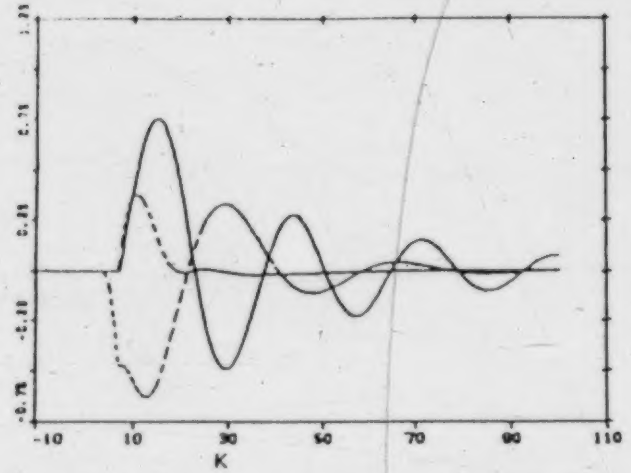


(a) $y_{ref}=(1\ 0)^T$



(b) $y_{ref}=(0\ 1)^T$

Figure 4.7 Output of the predictor in the distillation column example for multi-loop PI control and for setpoint changes; OR: continuous, IF: dashes



(a) $v=(1\ 0)^T$

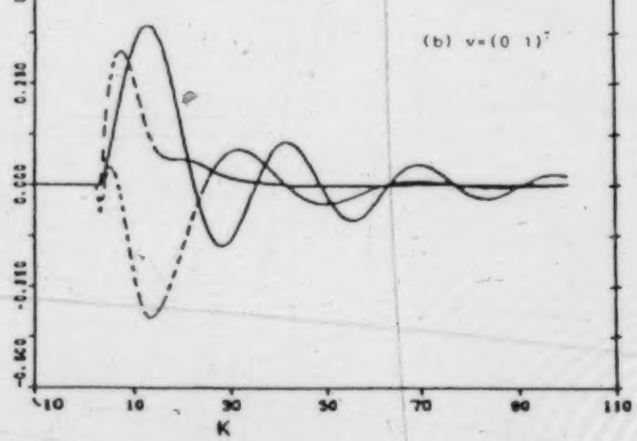
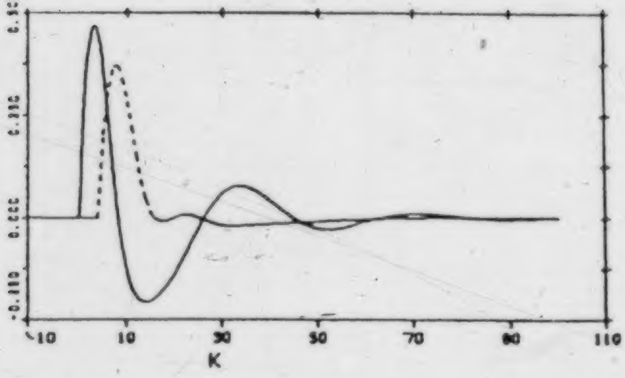
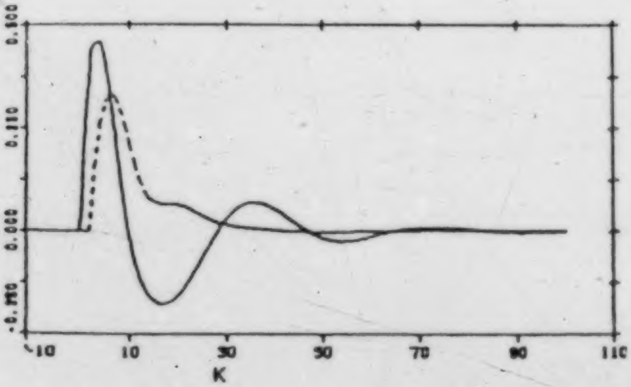


Figure 4.8 Simulated responses of the distillation column for delay compensated, multi-loop PI control and for disturbance changes; no compensation: continuous, OR: small dashes, IF: large dashes



(a) $v=(1\ 0)^T$



(b) $v=(0\ 1)^T$

Figure 4.9 Output of the predictor in the distillation column example under multi-loop PI control for disturbance changes; OR: continuous, IF: dashes

over the OR approach in the presence of disturbances. However, by designing the controller based on the plant output y (instead of y_p), a better control performance may be obtained for the plant with OR compensator. But then the design advantage of being able to work with a delay-free model in the OR scheme is lost.

For comparison, the closed-loop responses of the simulated plant without any delay compensation under multi-loop PI control were obtained and are shown in Figures 4.6 and 4.8. The controller parameters correspond to a well-tuned continuous PI controller reported in Wood and Berry (1973) and are presented in Table 4.3. The responses are quite oscillatory even at the low gain values used and entirely unsatisfactory. As compared with this, the IF approach results in significantly better performance. The OR approach generally gives better performance, but because of unpredictable interactions caused by removing all the delays it may sometimes give worse performance than ordinary PI control as in the present case (cf. Figure 4.6).

4.3.6 A recycle reactor example

A two stage, recycle chemical reactor train in which a simple 1st order reaction A→B is carried out was described in Ogunnaike and Ray (1979). A schematic diagram of the reactor is shown in Figure 4.10. All the flow rates are fixed and only the compositions are assumed to vary. The compositions of the feed to the two reactors are the manipulated variables. There is a separate disturbance feed to the first stage and its composition is the disturbance variable. The controlled outputs are the compositions of the

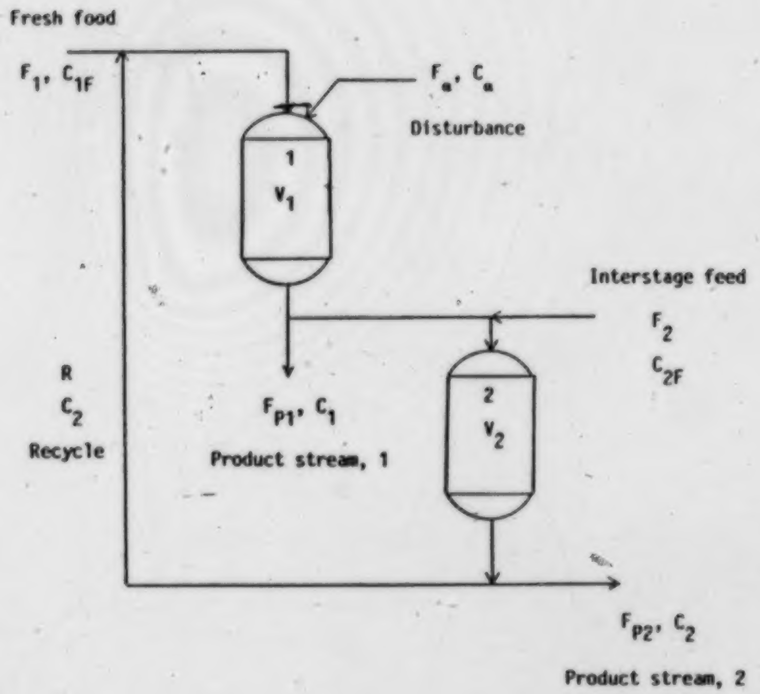

Fresh food
F_1, C_{1F}
F_a, C_a
Disturbance
1
V_1
Interstage feed
F_2
C_{2F}
R
C_2
Recycle
F_{P1}, C_1
Product stream, 1
2
V_2
F_{P2}, C_2
Product stream, 2

Figure 4.10 Schematic diagram of the two-stage chemical reactor

product stream from the two reactors.

Ogunnaike and Ray (1979) derived a dynamic model of the system as a continuous transfer function matrix assuming delays in the recycle stream. In the present example only measurement (output) and control (input) delays are considered to obtain a transfer function matrix in the form of equation (2). This does not result in any loss of generality as the transfer function matrices are usually derived from input/output data in practice.

From the continuous time model in Ogunnaike and Ray (1979), with a sample period $T_0=0.5$ units and the following parameters

$\mu=0.3$, $\theta_1=\theta_2=0.5$, $\nu_{11}=\nu_{21}=0$,

$D_{a_1}=D_{a_2}=0.5$, $\lambda_R=0.5$ and $\lambda_d=0.2$

plus a measurement delay of 2 and 3 sample intervals in the two outputs and a delay of 1 and 2 sample intervals in the two inputs, a discrete time model is obtained. This is presented in Table 4.1 (Example 5).

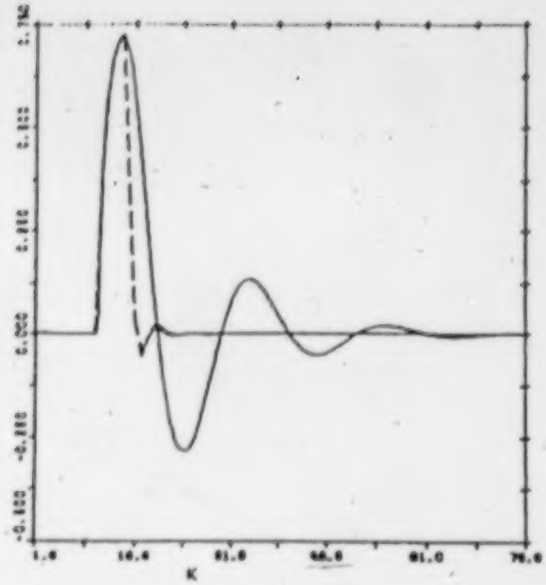
The inverse interactor matrix for this example is triangular and is given by

$$\xi_{T_m}(q^{-1}) := \begin{bmatrix} q^{-4} & 0 \\ .307q^{-5} & q^{-6} \end{bmatrix} \tag{31}$$

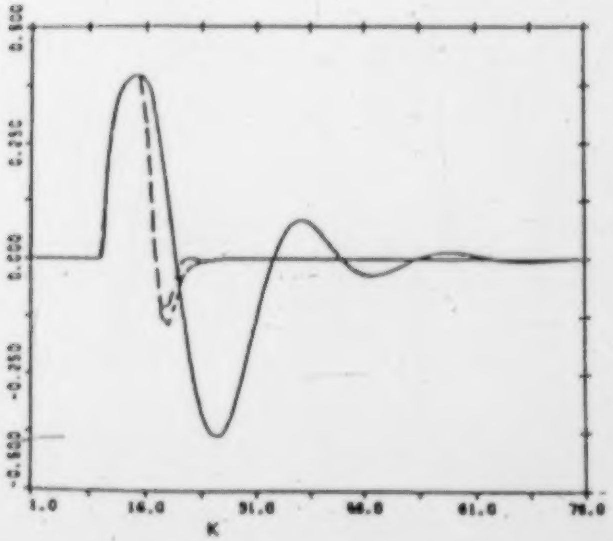
The predictors in the OR and IF approach are given in Table 4.2. Since the plant is characterized by low interactions both schemes result in approximately the same performance as shown by Figure 4.11 for a disturbance step change of v=5. The controller $T_c(q^{-1})$ is a diagonal PI controller with parameters given as in Table 4.4. The responses of the plant

Table 4.4 PI controller parameters for the reactor example

	Loop 1		Loop 2	
	q_0	q_1	q_0	q_1
OR approach	3.5	0	3.5	0
IF approach	3.5	0	3.5	0
No delay compensation	0.6	0	0.65	0



(a) Output 1 (i.e. y_1)



(b) Output 2 (i.e. y_2)

Figure 4.11 Simulated responses of the chemical reactor for delay compensated multi-loop PI control and for a disturbance change; no compesation: continuous, OR: small dashes; IF: large dashes

using PI controllers without any time-delay compensation are very oscillatory even at the relatively low gains used.

4.4 Feedforward control

The effect of measurable disturbances on the plant is often reduced by using a feedforward controller. Usually the configuration shown in Figure 4.12 is used which is independent of any feedback scheme that may simultaneously be acting on the plant. r is the measurable disturbance vector and $T_L(q^{-1})$ is the transfer matrix between r and the disturbance process output v. If the plant model is given and is denoted by $T_m(q^{-1})$ then the ideal feedforward controller is given by

$$T_C^{FF}(q^{-1}) = -T_m(q^{-1})^{-1}\, T_L(q^{-1}) \qquad (32)$$

so that $u^{FF} = T_C^{FF}(q^{-1})r$

If the plant model $T_m(q^{-1})$ contains delays then its inverse and hence the feedforward controller contains predictive terms and will be non-realizable. If the disturbance model also has delays then the feedforward controller will be causal (realizable) if the delays in $T_L(q^{-1})$ cancel the predictive terms in $T_m(q^{-1})^{-1}$. In general, the plant model and disturbance model may both contain multiple delays and it is often not immediately clear if the feedforward controller given by equation (32) will be causal or not. Also if the feedforward controller is non-causal, then the best design method for elimination of the predictive terms is still unanswered.

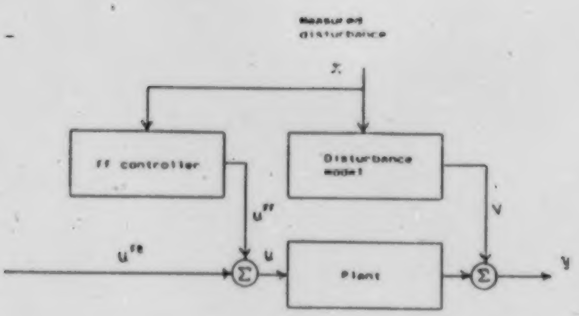

Measured
disturbance
z
ff controller
Disturbance
model
u^{ff}
V
u^{fb}
u
Plant
y
Σ
Σ

Figure 4.12 Feedforward controller

For discrete time systems, the inverse interactor factorization once again provides the most appropriate solution to the feedforward controller design problem. Equation (32) can be written as

$$T_C^{FF}(q^{-1}) = - R_{T_m}(q^{-1})^{-1}[\xi_{T_m}(q^{-1})^{-1}\xi_{T_L}(q^{-1})]R_{T_L}(q^{-1}) \tag{33}$$

The feedforward controller defined by equation (33) will be causal if and only if $[\xi_{T_m}(q^{-1})^{-1}\xi_{T_L}(q^{-1})]$ is finite at $q=\infty$. If $[\xi_{T_m}(q^{-1})^{-1}\xi_{T_L}(q^{-1})]$ is not finite at $q=\infty$ then the best (in the sense that only a minimum number of predictive terms need to be considered to achieve realizability) causal feedforward controller is obtained by setting the predictive terms of $[\xi_{T_m}(q^{-1})^{-1}\xi_{T_L}(q^{-1})]$ to unity. This is equivalent to setting any future prediction of the measured disturbance to its current value (i.e. $\hat{r}(t+i/t)=r(t)$) in Figure 4.12.

A physical interpretation of this design is that since the inverse interactor $\xi_{T_m}(q^{-1})$ extracts the minimum delays only minimum prediction is required in the feedforward controller, or equivalently maximum dynamic effects of the measured disturbances are compensated by $T_C^{FF}(q^{-1})$.

Some examples will illustrate these points more clearly. Consider Example 2 in Table 4.1. Since both $T_m(q^{-1})$ and $T_L(q^{-1})$ contain multiple delays, unless $T_C^{FF}(q^{-1})$ is computed it is difficult to see if it will be causal or not. The inverse interactor matrices for this example are,

$$\xi_{T_m}(q^{-1}) := \begin{bmatrix} q^{-3} & 0 \\ .75q^{-4} & q^{-5} \end{bmatrix} \tag{34}$$

and

$$\xi_{T_L}(q^{-1}) := \begin{bmatrix} q^{-4} & 0 \\ -.33q^{-4} & q^{-5} \end{bmatrix} \tag{35}$$

Since

$$[\xi_{T_m}(q^{-1})^{-1}\xi_{T_L}(q^{-1})] := \begin{bmatrix} q^{-1} & 0 \\ -.33q-.75 & 1 \end{bmatrix} \tag{36}$$

is not finite at $q=\infty$ (cf. q in (2,1) element), the feedforward controller will not be causal. A causal feedforward controller is obtained by setting $q=1$ in all the predictive terms in $[\xi_{T_m}(q^{-1})^{-1}\xi_{T_L}(q^{-1})]$, which results in

$$[\xi_{T_m}(q^{-1})^{-1}\xi_{T_L}(q^{-1})] := \begin{bmatrix} q^{-1} & 0 \\ -13/12 & 1 \end{bmatrix} \tag{37}$$

Substituting equation (37) into equation (33) gives

$$T_C^{FF}(q^{-1}) := \begin{bmatrix} -(32/30)q^{-1} & (20q^{-1}-32q^{-2})/45 \\ -39/45 & (-40-26q^{-1})/45 \end{bmatrix} \tag{38}$$

To illustrate the fact that the interactor factorization leads to compensating the maximum dynamic effects of the measured disturbance, consider the same example, but with the disturbance model $T_L(q^{-1})$ as given in Example 3 of Table 4.1. The IF approach in this case (note $\xi_{T_L}(q^{-1})=q^{-1}I$) leads to

$$T_c^{FF}(q^{-1}) := \begin{bmatrix} 3q^{-1}-3q^{-3} & 0 \\ -4q^{-1}+9q^{-4}-5q^{-5} & -8q^{-1}+3q^{-4}+5q^{-5} \end{bmatrix} \tag{39}$$

and the plant response is given by

$$y(t) := (1/\Delta)\begin{bmatrix} 3q^{-1}-3q^{-3} & 0 \\ -4q^{-1}+9q^{-4}-5q^{-5} & -8q^{-1}+3q^{-4}+5q^{-5} \end{bmatrix} v(t) \tag{40}$$

where $\Delta := (1-.8q^{-1})$

A standard design would proceed by designing $T_c^{FF}(q^{-1})$ according to equation (32) and setting the predictive terms to unity as before. This design leads to

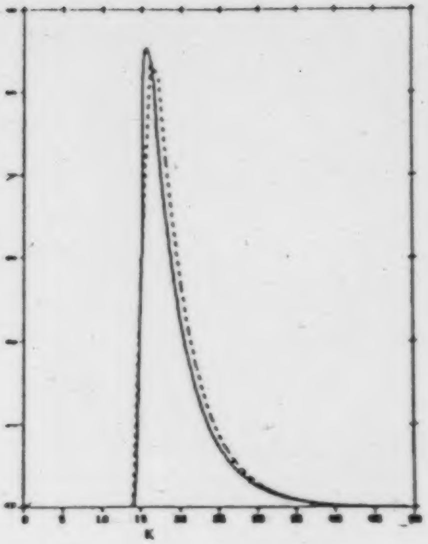
$$T_c^{FF}(q^{-1}) := \begin{bmatrix} -16/15 & -4/15 \\ -13/15 & -22/15 \end{bmatrix} \tag{41}$$

and the plant response is

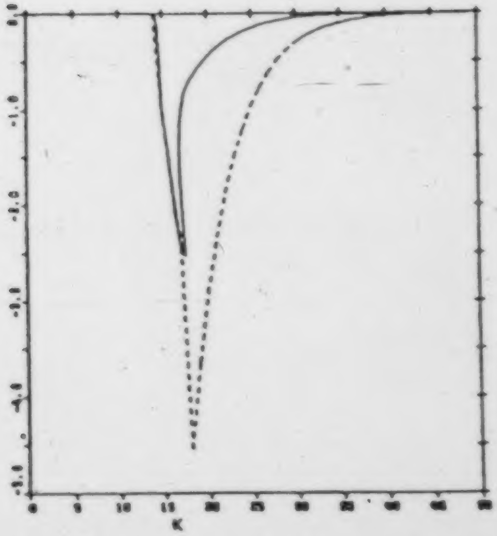
$$y(t) := (1/\Delta)\begin{bmatrix} 45q^{-1}-32q^{-3}-13q^{-4} & 30\ q^{-1}-8q^{-2}-22q^{-4} \\ -15q^{-1}-24q^{-4}+39q^{-5} & -60q^{-1}-6q^{-4}+66q^{-5} \end{bmatrix} \tag{42}$$

where $\Delta := 15(1-.8q^{-1})$

Clearly the feedforward controller in the standard design (cf. equation (41)) for this example compensates only the steady state effects of the measured disturbance. The simulated plant responses (cf. equations (40) and (42)) are shown in Figures 4.13 and 4.14. The IF design procedure

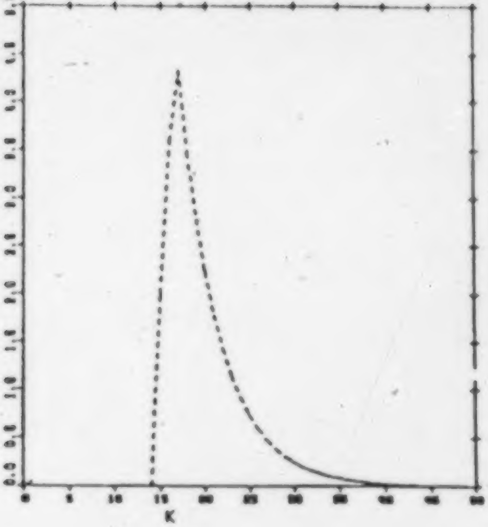


(a) Output 1 (i.e. y_1)

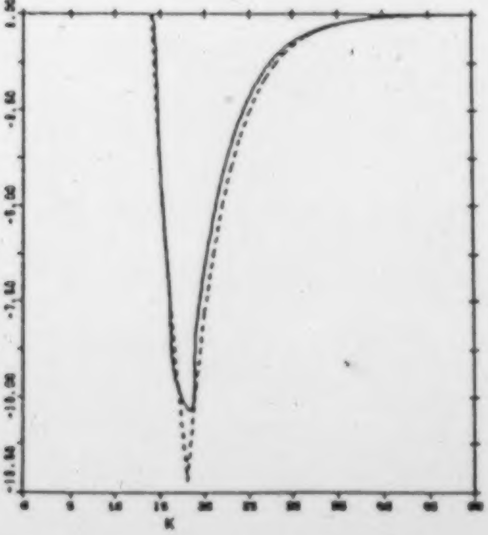


(b) Output 2 (i.e. y_2)

Figure 4.13 Simulated responses in the feedforward control example for a measured disturbance of $v=(1,0)^T$; IF: continuous, standard design: dashes



(a) Output 1 (i.e. y_1)



(b) Output 2 (i.e. y_2)

Figure 4.14 Simulated responses in the feedforward control example for a measured disturbance of $v=(0,1)^T$; IF: continuous, standard design: dashes

results in the maximum amount of dynamic compensation by $T_c^{FF}(q^{-1})$. The responses are therefore superior in the IF design procedure.

4.5 Model following

Knowledge of the portions of dynamical systems which remain invariant under a particular form of compensation is important in control systems analysis and synthesis, as for example in model following. Wolovich and Falb (1976) showed that the inverse interactor matrix is invariant under dynamic compensation. (For a pXl proper transfer function matrix T, any proper lxm transfer function matrix T_c is called a dynamic compensator of the system TT_c or T_cT.)

In a model following control system the objective is to make the plant output track the output of a reference model driven by a reference input. Feedback is applied to the plant so that the output of the closed-loop system when driven by the reference input is equal to the output of the model driven by the same reference input, i.e. $e(t) \to 0$ in Figure 4.15. Such model following systems occur in many practical situations, e.g. in placing the poles of a closed-loop system at desired locations. There are several ways in which a model following system can be designed (e.g. see Goodwin and Sin (1984)).

The inverse interactor is an invariant in a model following system. For discrete MIMO systems, the inverse interactor generalizes the concept of SISO delay. The model following or model matching question tries to answer whether or not there exists a causal compensator $T_c(q^{-1})$ such that the compensated plant, $T(q^{-1})T_c(q^{-1})$, follows a specified

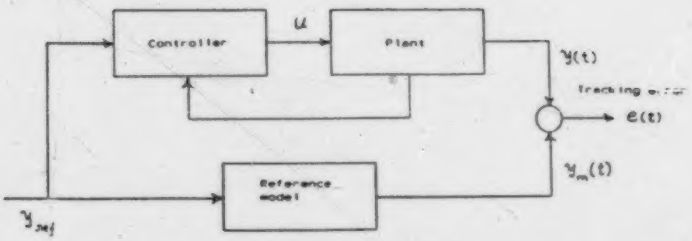

Controller
u
Plant
y(t)
Tracking error
e(t)
y_m(t)
Reference model
y_ref

Figure 4.15 A model following control system

model, $T_{ref}(q^{-1})$. A causal compensator, $T_C(q^{-1})$, (which in the discrete case implies that it does not contain any predictive terms) exists such that a given plant, $T(q^{-1})$, matches or follows a specified model, $T_{ref}(q^{-1})$, if and only if the model is of the form $T_{ref}(q^{-1})=\xi_{T_{ref}}(q^{-1})T_{ref_1}(q^{-1})$, for a specific transfer function matrix $T_{ref_1}(q^{-1})$ or equivalently $[\xi_T(q^{-1})^{-1}\xi_{T_{ref}}(q^{-1})]$ is finite at $q=\infty$.

This discussion once again supports the role of the inverse interactor as the most appropriate generalization of discrete time SISO system delay. For a delayed SISO plant in closed-loop with a causal controller, the delay is an invariant (i.e. the closed-loop must contain at least the plant delay). Likewise, for MIMO systems in closed-loop with a causal controller, the inverse interactor is an invariant.

4.6 MIMO predictive control

Predictive control schemes such as those based on the minimization of one-step and multi-step cost functions form a basis for a large class of adaptive predictive control strategies. Most predictive control schemes require and/or are based on a knowledge of the system inverse interactor matrix in the MIMO case. A lot of literature is available on this topic, e.g. see Goodwin and Sin (1984), Shah *et al.* (1986), Mohtadi *et al.* (1986) and Elliott and Wolovich (1984).

For example, Goodwin and Sin (1984) discuss how to obtain a simple one-step-ahead control scheme for a MIMO system. Suppose that a MIMO system is characterized by equation (3). Then a filtered reference output $\tilde{y}_{ref}$ is defined using

$$\hat{y}_{ref}(t) = \xi_T(q^{-1})^{-1}\ y(t) \tag{43}$$

Then the control signal u(t) is chosen so as to make

$$\hat{y}_{ref}(t) = \hat{y}(t) \tag{44}$$

where $\hat{y}(t)$ is a filtered or predicted system output given by

$$\hat{y}(t) = \xi_T(q^{-1})^{-1}\ y(t) \tag{45}$$

Since $\xi_T(q^{-1})$ is the "delay structure" of the MIMO system, $\xi_T(q^{-1})^{-1}y(t)$ defines the appropriate prediction of the output in analogy with the one-step-ahead SISO case. By assuming that the system transfer function matrix, $T(q^{-1})$, is given by $T(q^{-1})=\mathbf{A}(q^{-1})^{-1}\mathbf{B}(q^{-1})$, it can be shown that $\hat{y}(t)$ is related to $\{u(t)\}$ and $\{y(t)\}$ by a model of the form:

$$\hat{y}(t) = \alpha(q^{-1})\ y(t) + \beta(q^{-1})\ u(t) \tag{46}$$

where,
$\alpha(q^{-1}) := I + \alpha_1\ q^{-1} + \dots + \alpha_m\ q^{-m}$
$\beta(q^{-1}) := \beta_0 + \dots + \beta_{m+1}\ q^{-m}$,
β_0 is non-singular (follows from fact (2) of section 4.2.2). The proof requires solving a multivariable diophantine identity (see Goodwin and Sin (1984) for details).

The control-law is then obtained by setting

$$\hat{y}_{ref}(t)=\hat{y}(t)=\alpha(q^{-1})y(t) + \beta(q^{-1})u(t) \tag{47}$$

or

$$\mathbf{u}(t)=\beta_0^{-1}[\hat{y}_{ref}(t) - \alpha(q^{-1})\mathbf{y}(t) - (\beta(q^{-1})-\beta_0)\mathbf{u}(t)] \tag{48}$$

Such a controller, assuming perfect modeling of the plant and no disturbances, essentially achieves perfect open-loop setpoint tracking as illustrated by Figure 4.1(b).

In SISO adaptive predictive control, it is commonly assumed that the system delay is known. Analogously by assuming a knowledge of the inverse interactor, several people (e.g. Goodwin *et al.* (1980), Dugard *et al.* (1983, 1984), Elliott and Wolovich (1984) and Johansson (1982)) have developed globally stable adaptive control schemes, both implicit and explicit. The assumption that $\xi_T(q^{-1})$ is known *a priori* is reasonable if $\xi_T(q^{-1})=q^{-d}I$ or $\xi_T(q^{-1}) :=$ $\mathrm{diag}(q^{-d_1}, \ldots, q^{-d_p})$. But in general the inverse interactor is a lower triangular polynomial matrix and may contain real coefficients. It is therefore unreasonable to expect it to be known *a priori* as the knowledge of the inverse interactor is tantamount to the knowledge of full system transfer function matrix. Some recently proposed schemes such as those based on the minimization of multi-step cost functions avoid the requirement for complete knowledge of the inverse interactor matrix, see e.g. Shah *et al.* (1986) and Chapter 3 on MOCCA and Chapter 6 on MAPC.

4.7 Conclusions

The main conclusion of this work is that interactor factorization is an important and valuable technique that can be widely applied to the analysis and design of control schemes for MIMO systems with time-delays. More specific results are:

1) When the factorization $T(q^{-1})=\xi_T(q^{-1})R_T(q^{-1})$ results in a *diagonal* inverse interactor matrix, $\xi_T(q^{-1})$, then the delays defined by the diagonal inverse interactor matrix are the *natural* delays inherent in the process. They are obviously the delay between the open-loop response of $R_T(q^{-1})$ versus $T(q^{-1})$ but are also the delay between the closed-loop response of a Smith predictor type of feedback control system based on $R_T(q^{-1})$ versus the response of $T(q^{-1})$. For example, if explicit prediction were used, i.e. estimate $y_i(t+d_i)$ at time t, then the interactor factorization provides the set of minimal values of d_i. In this sense, interactor factorization provides a natural extension of SISO concepts.
2) Interactor factorization clarifies and extends feedback, time-delay compensation techniques such as those presented by Smith (1957), Ogunnaike and Ray (1979), Jerome and Ray (1986) etc. For a significant class of problems, it can be shown to provide the "optimal" solution and simulation results show that the improvement over previous methods can be very significant.
3) For feedforward control applications, interactor factorization: provides a formal basis for determining the causality of the controller; if prediction $\hat{r}(t+i/t)$ of the measured disturbance, r, is to be used it results in the minimum predictions, i; ensures that if the controller is non-causal, and if the controller is

designed based on the "assumption" that $\hat{r}(t+i/t)=r(t)$, $i \geq 1$, then $R_T(q^{-1})$ provides the maximum dynamics for inclusion in the controller. These advantages were confirmed by simulated results.

4) For a causal model following system, the time-delays defined by the inverse interactor matrix are an "invariant" of the system in the sense that the output of the system will lag the reference input(s) by "at least" these values.

5) The role of the inverse interactor matrix in MIMO predictive control and MIMO adaptive predictive control is pointed out but not treated in detail.

References

Alevisakis, G. and D. E. seborg (1973): "An Extension of the Smith Predictor Method to Multivariable Linear Systems Containing Time-delays", Intl. J. Control, 3, 541.

Dahlin, E. B. (1967): "Designing and Tuning Digital Controllers", Instruments and Control Systems, 41, 6, 77.

Dugard L., G. C. Goodwin and C. E. deSouza (1983): "Prior Knowledge in Model Reference Adaptive Control of Multi-input Multi-output Systems", IEEE CDC, San Antonio, TX, USA, 455.

Dugard, L., G. C. Goodwin and X. Xianya (1984): "The Role of the Interactor Matrix in Multivariable Stochastic Adaptive Control", Automatica, 20, 5, 701.

Elliott, H. and W. A. Wolovich (1984): "Parameterization Issues in Multivariable Adaptive Control", Automatica, 20, 5, 533.

Garcia, C. E. and M. Morari (1983): "Internal Model Control 2: Design Procedure for Multivariable Systems", I&EC Process Des. and Dev., 472.

Goodwin, G. C. and R. S. Long (1980): "Generalization of Results on Multivariable Adaptive Control", IEEE Trans. on Aut. Con., AC-25, 6, 1241.

Goodwin, G. C., P. J. Ramadge and P. E. Caines (1980): "Discrete Time Multivarible Adaptive Control", IEEE Trans. Aut. Con., AC-25, 4, 449

Goodwin, G. C. and K. S. Sin (1984): Adaptive Filtering, Prediction and Control, Prentice-Hall.

Jerome, N. F. and W. H. Ray (1986): "High-Performance Multivariable Control Strategies for Systems Having Time Delays", AIChE Journal, 32, 6, 914.

Johansson, R. (1982): "Parametric Models of Linear Multivariable Systems for Adaptive Control", IEEE CDC, 989.

Mohtadi, C., S. L. Shah and D. W. Clarke (1986): "Generalized Predictive Control of Multivariable Systems", Report No. OUEL 1640/86, Univ. of Oxford.

Ogunnaike, B. A. and W. H. Ray (1979): "Multivariable Controller Design for Linear Systems having Multiple Delays", AIChE J., 25, 6, 1043.

Singh, R. P. and K. S. Narendra (1984): "Prior Information in Design of Multivariable Adaptive Controllers", IEEE Trans. Aut. Control, **AC-29**, 12, 108.

Shah, S. L., C. Mohtadi and D. W. Clarke (1986): "Multivariable Adaptive Control without a Prior Knowledge of the Delay Matrix", O.U.E.L. Technical Report.

Smith, O. J. M. (1957): "Closer Control of Loops with Dead Time", Chem. Engg. Prog., **53**, 217.

Wolovich, W. A. and P. L. Falb (1976): "Invariants and Canonical Forms under Dynamic Compensation", SIAM J. Control and Optimization, **14**, 6, 996.

Wolovich, W. A. and H. Elliott (1983): "Discrete Models for Linear Multivariable Systems", Intl. J. Control, **38**, 2, 337.

Wood, R. K. and M. W. Berry (1973): "Terminal Composition Control of a Binary Distillation Column", Chem. Engg. Sci., **28**, 1707.

Appendix 4.1

Procedure for obtaining the interactor matrix

Given an mxm, proper transfer function matrix, $G(q)$, in the forward shift operator, q, there exists a unique interactor matrix, $\xi_G(q)$ such that

$$\lim_{q\to\infty} \xi_G(q)G(q) = K_G$$

where K_G is finite and non-singular; $\xi_G(q):=H_G(q)\mathrm{diag}(q^{f_1},\dots,q^{f_m})$; and

$$H_G(q)=\begin{bmatrix} 1 & 0 & \dots & 0 \\ h_{21}(q) & 1 & \dots & 0 \\ h_{31}(q) & h_{32}(q) & 1 \;.. & 0 \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ h_{m1}(q) & \dots & \dots & 1 \end{bmatrix}$$

One can always write $G(q)=R(q)P(q)^{-1}$ where $R(q)$ and $P(q)$ are relatively prime polynomial matrices. Let $r:=\partial(\det\{R(q)\})$ be the degree of the determinant of $R(q)$; and $n:=\sum_{i=1}^{m} d_i$ where d_i is the i^{th}-column degree of $P(q)$.

Step 1

There are unique integers μ^i, i=1, 2, ..., m such that

$$\lim_{q\to\infty} q^{\mu^i} G_i(q)=\tau_i, \quad i=1,\ 2,\ \ldots,\ m$$

where $G_i(q)$ is the i^{th} row of $G(q)$ and τ_i is both finite and non-zero. Then the first row, $\xi_G(q)_1$ of $\xi_G(q)$ is defined by

$$\xi_G(q)_1=(q^{\mu^1},\ 0,\ 0,\ \ldots,\ 0)$$

so that $\lim_{q\to\infty} \xi_G(q)_1 G(q)=\xi_1=\tau_1$.

Step 2

If τ_2 is linearly independent of ξ_1 then one sets

$\xi_G(q)_2 = (0,\ q^{\mu^2},\ 0,\ \ldots,\ 0)$

so that $\lim_{q\to\infty} \xi_G(q)_2 G(q)=\xi_2=\tau_2$. Otherwise (i.e. if τ_2 and ξ_1 are linearly dependent) let $\tau_2=\alpha_1^{\ 1}\xi_1$ with $\alpha_1^{\ 1}\neq 0$. Then

$$\tilde{\xi}_G^{\ 1}(q)_2 = q^{\mu_2^1}((0,\ q^{\mu^2},\ 0,\ \ldots,\ 0) - \alpha_1^{\ 1}\xi_G(q)_1)$$

where $\mu_2^{\ 1}$ is a unique integer such that $\lim_{q\to\infty} \tilde{\xi}_G^{\ 1}(q)_2 G(q)=\tilde{\xi}_2^{\ 1}$ is both finite and non-zero. If $\tilde{\xi}_2^{\ 1}$ is not linearly dependent on ξ_1 then

$$\xi_G(q)_2 = \tilde{\xi}_G^{\ 1}(q)_2$$

and $\lim_{q\to\infty} \xi_G(q)_2 G(q)=\xi_2=\tilde{\xi}_2^{\ 1}$. ξ_1 is linearly independent of ξ_2. If $\tilde{\xi}_2^{\ 1}$ is not linearly independent of ξ_1 then $\tilde{\xi}_2^{\ 1}=\alpha_1^{\ 2}\xi_1$ and therefore let

$$\tilde{\xi}_G^{\ 2}(q)_2=q^{\mu_2^2}(\tilde{\xi}_G^{\ 1}(q)_2-\alpha_1^{\ 2}\xi_G(q)_1)$$

where μ_2^2 is a unique integer such that $\lim_{q\to\infty} \tilde{\xi}_G^2(q)_2 G(q)=\tilde{\xi}_2^2$ is both finite and non-zero.

The above procedure is repeated until linear independence is observed, or $\mu_1+\mu_2^k=n-r$. If $\mu_1+\mu_2^k=n-r$, then set $f_3=f_4=\ldots=f_m=0$ and the corresponding $h_{ij}(q)=0$.

The remaining rows of $\xi_G(q)$ are defined recursively in an entirely analogous manner. So finally one gets either i) the complete interactor; or ii) $\xi_G(q)_1$, $\xi_G(q)_2$, ..., $\xi_G(q)_r$, $r\leq m$ such that $\lim_{q\to\infty} \xi_G(q)_j G(q)=\xi_j$, for $j\leq r$, with ξ_1, ξ_2, ..., ξ_r linearly independent and $\sum_{i=1}^{r} f_i=n-r$. In case (ii) one sets $f_{r+1}=\ldots=f_m=0$ and the corresponding $h_{ij}(q)=0$ to obtain $\xi_G(q)$. If r=m then

$\lim_{q\to\infty} \xi_G(q)G(q)=(\xi_1^T\ldots\xi_m^T)^T=K_G$ is finite and non-singular.

Some examples to illustrate the above procedure

Example 1: Consider

$$G(q) = \begin{bmatrix} \frac{1}{q-.1} & \frac{1}{q-.2} \\ \\ \frac{2}{q-.3} & \frac{2}{q-.4} \end{bmatrix}$$

Clearly $\mu_1=1$ and $\mu_2=1$ so that $\tau_1=(1,1)$ and $\tau_2=(2,2)$. Therefore, $\xi_G(q)_1=(q^{\mu_1},0)=(q,0)$ and $\lim_{q\to\infty} \xi_G(q)_1 G(q)=\xi_1=\tau_1=(1,1)$. Since ξ_1 and τ_2 are linearly dependent, one takes $\tau_2=a_1^1\xi_1=(2,2)$ with $a_1^1=2$.

$\tilde{\xi}_G^1(q)_2=q^{\mu_2^1}((0,q)-2(q,0))$

Here $\mu_2^1=1$ so $\tilde{\xi}_G^1(q)_2=q(-2q,q)=(-2q^2,q^2)$ and
$\lim_{q\to\infty} \tilde{\xi}_G^1(q)_2 G(q)=\tilde{\xi}_2^1=(0.4,0.4)$
Once again, $\tilde{\xi}_2^1$ and ξ_1 are linearly dependent and $\tilde{\xi}_2^1=a_1^2\xi_1$

with $\alpha_1^2=0.4$; so let

$$\tilde{\xi}_G^2(q)_2=q^{\mu_2^2}((-2q^2,q^2)-0.4(q\ 0)) \text{ where } \mu_2^2=1.$$

Therefore,

$\tilde{\xi}_G^2(q)_2=q(-2q^2-.4q,q^2)=(-2q^3-0.4q^2,q^3)$ and

$\lim_{q\to\infty}\tilde{\xi}_G^2(q)_2G(q)=(0.12,0.16)=\tilde{\xi}_2^2$ is linearly independent of ξ_1.

Therefore,

$$\xi_G(q)_2=\tilde{\xi}_G^2(q)_2=(-2q^3-0.4q^2,q^3)$$

so that the interactor is given by

$$\xi_G(q) = \begin{bmatrix} q & 0 \\ -2q^3-0.4q^2 & q^3 \end{bmatrix}$$

and $\lim_{q\to\infty}\xi_G(q)G(q)=K_G=\begin{bmatrix} 1 & 1 \\ .12 & .16 \end{bmatrix}$

is finite and non-singular.

Example 2: Consider

$$G(q) = \begin{bmatrix} \dfrac{1}{q^3+4q^2} & \dfrac{2}{q^4-0.5q^3} \\ \\ \dfrac{3}{q^4+.1q^3} & \dfrac{1}{q^5+0.8q^4} \end{bmatrix}$$

Clearly $\mu_1=3$, $\mu_2=4$, $\tau_1=(1,0)$ and $\tau_2=(3,0)$.

So $\xi_G(q)_1=(q^{\mu_1},0)=(q^3,0)$

and $\lim_{q\to\infty}\xi_G(q)_1G(q)=\xi_1=\tau_1=(1,0)$.

Since ξ_1 and τ_2 are linearly dependent, $\tau_2=\alpha_1^1\xi_1$ where $\alpha_1^1=3$.

One therefore takes

$$\tilde{\xi}_G^1(q)_2=q^{\mu_2^1}((0,q^4)-3(q^3,0)) =q^{\mu_2^1}(-3q^3\ q^4).$$

Here $\mu_2{}^1=1$ so $\tilde{\xi}_G{}^1(q)_2=(-3q^4,q^5)$

$\lim_{q\to\infty}\tilde{\xi}_G{}^1(q)_2G(q)=(0.9 \quad -7)$ is linearly independent of ξ_1. Therefore, $\xi_G(q)_2=\tilde{\xi}_G{}^1(q)_2=(-3q^4 \ q^5)$ and the interactor is given by $\xi_G(q) = \begin{bmatrix} q^3 & 0 \\ -3q^4 & q^5 \end{bmatrix}$

$$K_G = \begin{bmatrix} 1 & 0 \\ .9 & -7 \end{bmatrix}$$

is both finite and non-singular.

5. Improved least squares identification

5.1 Introduction

Parameter estimation techniques are widely used in open-loop applications such as process modeling and in closed-loop environments such as adaptive control. Several excellent review articles and textbooks have been written over the past few decades (for example, Astrom and Eykhoff (1971), Nieman (1971), Iserman (1980), Strejc (1980, 1981), Ljung and Soderstrom (1983), and Sinha and Kuszta (1983)).

Numerous different parameter estimation techniques have been developed and they each have advantages, disadvantages and particular target areas for application, but there is little doubt that the most widely used estimation technique is least squares. However, inspite of its wide spread use, the basic least squares algorithm has a number of potential problems, some of which are dealt with in this chapter. The rest of this Introduction Section indicates *what* some of the main problems are and very briefly indicates *why* and *when* they occur. Section 5.2 contains a step by step description of the proposed Improved Least Squares (ILS) algorithm which is designed to deal with these problems. The theoretical properties of ILS are defined in Section 5.3 and this is followed by a literature review plus discussion of ILS properties and performance. Simulated examples are presented to demonstrate the various problems with the standard RLS algorithm and the solutions proposed in ILS. These examples

---

A version of this chapter was presented at the American Control Conference (received 'best presentation in session' award): N. R. Sripada and D. G. Fisher, 1987, Proc. ACC, Minnesota; and has also been published: 1987, Int. J. Control, 46, 6, 1889.

are summarized in Appendix 5.2 and discussed in the chapter where appropriate.

5.1.1 Turn-off

With standard least squares (cf. equations (6), (7) and (8) with $\lambda(t)=1\ \forall t$), estimation starts with the initial values of the parameter estimates $\hat{\theta}(0)$, and covariance $P(0)$. As estimation proceeds, assuming sufficient excitation, the covariance, $P(t)$, decreases until parameter updating (equation (6)) turns off. A poor choice of initial conditions could mean that estimation stops before the parameter estimates converge. Selecting a forgetting factor, $\lambda(t)$, less than 1 in equation (8) will prevent $P(t) \rightarrow 0$ and avoid "turn-off". Example 1 in Appendix 5.2 illustrates the turn-off phenomenon.

5.1.2 Covariance blowup

If the forgetting factor $\lambda(t)=\lambda < 1$ in equation (8) then $P(t)$ can become very large, i.e. covariance "blowup" or "windup" occurs. This is easiest to see for the special case where the process is operating at steady state and all elements of the regressor vector are zero. Equation (8) then reduces to $P(t) = P(t-1)/\lambda(t)$ and it is obvious that the magnitude of $P(t)$ will increase with time. Authors such as Fortescue *et al.* (1981) have proposed a variable forgetting factor which avoids this problem in deterministic systems. Note however that covariance blowup may still occur in stochastic systems during periods of steady state operation. A large covariance matrix makes the algorithm very sensitive to even small perturbations in the regressor vector and hence can cause large changes in the estimated parameters and in system performance. Covariance blowup is demonstrated by Examples 2, 3 and 4 in Appendix 5.2.

5.1.3 Parameter drift and bursting

If conditions arise such that the covariance matrix in equation (6) becomes fairly large and the regressor vector, $\phi(t)$, is dominated by (random) noise then it is obvious that the parameter estimates, $\hat{\theta}(t)$, are unlikely to move in a direction that improves the input/output model. These parameter changes are referred to as drift. It has been observed in adaptive control applications and simulation studies (e.g. Anderson (1985)) that the parameters can drift into an unstable region. The result is large perturbations or "bursting" in the process input/output variables. The rich excitation caused by the bursting usually results in improved parameter estimation so that the bursting is self-correcting. Note that covariance blowup increases the probability of parameter drift and hence of bursting but by itself is not the problem. Parameter drifting and bursting are demonstrated in Examples 3, 4 and 9 in Appendix 5.2.

5.1.4 Excitation

Excitation is required for parameter estimation. If a strong, persistently exciting input signal is present then (as discussed later) the result is normally rapid convergence. In the absence of excitation stochastic systems may exhibit parameter drift (Anderson (1985)), and/or covariance blowup as discussed above. Some authors recommend adding an external excitation signal but this is often inappropriate or inconvenient. An alternative is to turn off the parameter estimation when excitation is insufficient. The algorithm proposed in this chapter emphasises the latter approach but permits both. Example (9) in Appendix 5.2 illustrates the problems associated with lack of excitation.

5.1.5 Biased estimates

Biased parameter estimates can arise, for example, when the process output elements in the regressor vector are biased due to the presence of an external signal. The external signal may be a constant or slowly changing bias or an independent external signal such as noise or process disturbances. Techniques such as extended least squares (ELS) have been developed to handle such situations but are generally computationally more demanding and give slower parameter convergence than basic least squares. The proposed algorithm focuses on removing the slowly changing (d.c.) bias but is written so that it can also be used for ELS. Consideration is also given to model-process mismatch (MPM) caused by process non-linearities and/or reduced order models. The effect of a d.c. bias on the estimated parameters and the effect of MPM due to reduced order modeling are illustrated by Examples 5 and 8 of the Appendix 5.2.

5.1.6 Theoretical properties

In general, a good estimation method should have proven theoretical properties (e.g. an exponential rate of convergence to the true parameter values), the ability to track slowly changing process parameters, restriction of the parameter estimates to *a priori* known convex regions etc. Many proposed modifications to the basic least squares algorithm (such as covariance resetting) alter the geometry and convergence due to the true least squares property. The proposed ILS algorithm retains the theoretical properties of least squares.The most important properties of ILS are summarized in Theorems 1 and 2 and proven in Appendix 5.1.

5.1.7 Numerical properties

Estimation algorithms are normally implemented on digital computers and hence there is the possibility of numerical ill-conditioning. Examples (6) and (7) in Appendix 5.2 illustrate typical numerical problems caused by an ill-conditioned covariance matrix in the RLS algorithm and ILS with constant trace. The proposed algorithm uses data preprocessing plus internal checks on numerical conditioning to improve the performance of the algorithm without destroying the desirable theoretical properties.

Many other problems and properties of least squares have been discussed in the literature but the above should be sufficient to indicate the motivation and directions for improving least squares estimation. Additional points are brought out during the development of the improved algorithm and/or in the discussion of the literature in Section 5.4.

5.2 Improved least squares estimation

After presenting the assumed process representation and summarizing the basic least squares algorithm, the proposed improved least squares estimation algorithm is defined and discussed in a stepwise fashion. It should be emphasized that the integration and interaction of the different steps is as important as any single feature taken individually.

5.2.1 Process representation

For purposes of this presentation it is assumed that the process can be represented in the form:

$$Y(t) = G_0\,U(t) + G_1\,e(t) + d'(t) + \xi'(t) \qquad (1)$$

where U, Y, e, d' and ξ' are the plant input, output, a second external perturbation input (e.g. noise), deterministic disturbances or bias, and plant modeling error respectively. The plant input/output relationship G_0 and perturbation transfer function G_1 are assumed to be rational functions of the form:

$$G_0 = \frac{B}{A} \text{ and } G_1 = \frac{C}{A}$$

where A, B and C are polynomials in the backward shift operator q^{-1}. Note that for applying (recursive) parameter estimation techniques such as least squares a model that is linear in the parameters is required. A common denominator is assumed without loss of generality. The signal d' is assumed to be a constant or a slowly time-varying d.c. bias. Multiplying both sides of (1) by A gives

$$A\,Y(t) = B\,U(t) + Ce(t) + A\,d'(t) + A\,\xi'(t) \qquad (2)$$

or

$$A\ Y(t) = B\ U(t) + C\ e(t) + d(t) + \xi(t) \tag{3}$$

Again for simplicity it is assumed that $C = 1$. If $C \neq 1$, then ELS (Extended Least Squares) or GLS (Generalized Least Squares) can be used. The same ILS algorithm can be used for ELS by proper reformulation of the regressor and parameter vectors.

The d.c. bias d can be eliminated from equation (3) by using the zero mean deviations $y(\cdot) := (Y(\cdot) - Y_{d.c.})$ and $u(\cdot) := (U(\cdot) - U_{d.c.})$ of $Y(\cdot)$ and $U(\cdot)$ where $Y_{d.c.}$ and $U_{d.c.}$ are the mean values of $Y(\cdot)$ and $U(\cdot)$. The d.c. (or mean) values may be known *a priori*, calculated as a function of measured process values, or estimated on-line as discussed later. Then equation (3) becomes

$$A\ y(t) = B\ u(t) + e(t) + \xi(t) \tag{4}$$

The $e(t)$ and $\xi(t)$ terms, both of which usually represent high-frequency signals, can be lumped together into one error term $\eta(t)$. If $A := 1 + \sum_{i=1}^{n} a_i q^{-i}$ and $B := (\sum_{i=1}^{m} b_i q^{-i}) q^{-\tau-1}$ where n and m are assumed or known upperbounds on the orders of the A and B polynomials, and τ is the known process delay in number of sample intervals, then the model of the real process becomes:

$$y(t) = \theta_*^T \phi(t) + \eta(t) \tag{5}$$

with the true parameter values $\theta_* := (a_1, \ldots, a_n, b_1, \ldots, b_m)^T$

and $\phi(t) := (-y(t-1), \ldots -y(t-n), u(t-\tau-1), \ldots u(t-\tau-m))^T$. Note that without loss of generality it is possible to use $n=m$.

5.2.2 Basic least squares algorithm

The basic recursive least squares algorithm to estimate the parameters of equation (5) is given by:

$$\hat{\theta}(t) = \hat{\theta}(t-1) + K(t)(y(t) - \hat{\theta}(t-1)^T\phi(t)) \tag{6}$$

$$K(t) = P(t)\phi(t) \tag{7}$$

$$P(t) = \left(I - \frac{P(t-1)\phi(t)\phi(t)^T}{\lambda(t) + \phi(t)^T P(t-1)\phi(t)}\right) \frac{P(t-1)}{\lambda(t)} \tag{8}$$

where $\hat{\theta}(0)$ is pre-specified and $P(0)$ is any positive definite matrix. This algorithm minimizes the following quadratic loss function

$$V(\hat{\theta}) := \sum_{i=0}^{t} \left[\prod_{j=i+1}^{t} \lambda(j)\right] (y(i) - \hat{\theta}^T\phi(i))^2 + V^0(\hat{\theta})$$

with the convention that $[\prod_{j=i}^{t} \lambda(j)]$ is equal to the product $\lambda(i)\lambda(i+1) \ldots \lambda(t)$ when $i \leq t$ and to 1 when $i > t$. Here $V^0(\hat{\theta}) := (\hat{\theta} - \hat{\theta}(0))^T P(0)^{-1}(\hat{\theta} - \hat{\theta}(0))$ and is due to the effect of the initial conditions. A proof of this statement is given in (Goodwin and Sin (1984)). In the standard recursive least squares algorithm, $\lambda(t)=1$ $\forall t$. If $\lambda(t)=\lambda \in (0,1)$ then the algorithm becomes an exponentially weighted least squares algorithm and $\lambda(t)$ is referred to as an exponential forgetting factor.

5.2.3 Formulation of the improved least squares algorithm

Each step of the proposed algorithm is discussed below. It is assumed that the following information is passed to the ILS algorithm at time, t: $\hat{\theta}(t-1)$, $\{Y(t)\}$, $\{U(t)\}$, $\hat{Y}_{d.c.}(t-1)$, $\hat{U}_{d.c.}(t-1)$, $P(t-1)$, $P(t-1)^{-1}$, $S(t-1)$ plus the user specified parameters: κ, k (a scalar multiplier of the right hand side of equation (8)), ι, $\lambda_y(t)$ and $\lambda_u(t)$.

Step 1, d.c. value elimination or high-pass filtering: The d.c. values of $\{U(\cdot)\}$ and $\{Y(\cdot)\}$ can be estimated independently of the process parameters using the least squares procedure (e.g. see Young (1984)). The simplified equations for estimating the d.c. values are:

$$\hat{Y}_{d.c.}(t) = \lambda_y(t)\,\hat{Y}_{d.c.}(t-1) + (1 - \lambda_y(t))\,Y(t) \tag{9}$$

$$\hat{U}_{d.c.}(t) = \lambda_u(t)\,\hat{U}_{d.c.}(t-1) + (1 - \lambda_u(t))\,U(t-1) \tag{10}$$

These equations can also be interpreted as "exponential filters" which exponentially forget old data or, equivalently, determine the contribution of past data (cf. window) to the filtered value. Setting the forgetting factors $\lambda_y(t)$ and $\lambda_u(t)$ to values less than unity allows tracking of slowly time-varying mean values and therefore a slowly time-varying d.c. bias. One option is to select $\lambda_y(t) = \lambda_u(t) = \lambda(t)$ which means that the parameter estimation and mean value estimation are done approximately over the same window of data. However, it is recommended that Λ_u and Λ_y be specified separately. Note that as $\lambda_y(t)$ or $\lambda_u(t)$ -> 1 the estimated mean becomes constant, e.g. the initial value. However, it is possible or even desirable in some instances to make the forgetting factors, λ_u and λ_y, a function of plant conditions (e.g. U and Y). It is also conceivable to use a CUSUM test (e.g. see Gilchrist (1977), Woodall (1986) and Chapter 6) to detect sudden level changes in U and Y and to set the values of λ_u and λ_y appropriately. The effect of d.c. bias elimination is demonstrated by Example 10 in Appendix 5.2. Compare this with Example 5 where not taking the d.c. bias into account results in an offset in the estimated parameters.

Step 2, Low-pass filtering: The plant modeling error, $\xi(t)$, usually contains high frequency modes and its effect can be reduced by low-pass filtering of the data (i.e. both $\{U(\cdot)\}$ and $\{Y(\cdot)\}$ to get $\{U_f(\cdot)\}$ and $\{Y_f(\cdot)\}$). This low-pass filtering may be in addition to any anti-aliasing, analog pre-filtering used on the output prior to sampling. Note that the sampling operation limits the high frequency content appearing in the sampled output. The band-width of the low-pass filter used for filtering the data must be large enough to permit accurate estimation (tracking) of the actual variations in the process parameters. If the bandwidth of the low-pass filter is too low, then in the limit, only variations in the process gain are tracked. The problems caused by model process mismatch illustrated in Example 8 of Appendix 5.2 are to some extent eliminated by low-pass filtering of the data as shown by Example 13.

Low-pass filtering can also be used to eliminate the structure in the noise in equation (3) to obtain unbiased estimates of A and B. Thus if $C \neq 1$ then a user specified approximation of C can be used to filter the input/output data (e.g. if T is an approximation of C, then $Y_f(\cdot):=\frac{1}{T(q^{-1})}Y(\cdot)$). If T=C, the noise effect is completely eliminated on the estimated A and B. The parameters of T can also be estimated in a separate step. This procedure is adopted in GLS. Note that the use of the T filter is an alternative to ELS.

Step 3, Normalization: The regressor vector $\phi(t)$ of low-pass filtered, zero-mean deviations, i.e. $\{u(\cdot)\}$ and $\{y(\cdot)\}$ where $u(\cdot) := U_f(\cdot)-U_{d.c.}$ and $y(\cdot) := Y_f(\cdot)-Y_{d.c.}$, are normalized by $n(t) := \max(1, \|\phi(t)\|)$ so that all the elements of $\phi_n(t) := \phi(t)/n(t)$ are ≤ 1. Normalization is required in proving the convergence of the estimation scheme (cf. Theorem 2, Section 5.3). It also improves the numerical quality of the computations.

Step 4, Data scaling: It is well known that in practical applications of least squares estimation poor data often lead to numerical problems. This is especially true if the excitation of the process is poor. One way to make the algorithm numerically more robust is to use a scheme such as square-root filtering or UDU^T-factorization (e.g. see Ljung and Soderstrom 1983) in the updating of P. An alternative is proposed here based on the idea of scaling which appears to have certain advantages. The proposed scaling procedure can also be used in conjunction with a square-root version of the algorithm. This can save some computation because the scaling matrix is computed on the basis of a square-root of P as discussed below.

Consider the problem of solving the set of linear equations Ax=b. A robust way of doing this is to solve the transformed set of equations, $A'x'=b'$ where $A' := PAQ$, $x' := Q^{-1}x$ and $b' := Pb$, and recover the original solution, x, from $x=Qx'$. Here P and Q are diagonal scaling matrices. The advantage of this approach is that the transformed set of equations may be numerically better conditioned than the original set of equations, i.e. the condition number $C\{A'\}$ of the transformed matrix A' may be less than or equal to the condition number $C\{A\}$ of the original matrix A. Therefore the computed solution, $x=Qx'$, obtained by using the scaling transformation would be less sensitive to (numerical) errors in b and/or A. In fact, the scaling matrices P and Q can be calculated (Noble (1969)) such that the condition number of A', $C\{A'\}$, is *minimized*. Since the least squares method is essentially solving a set of linear equations of the above type where $A := \Phi(t)^T\Phi(t)$, $x := \theta$ and $b := \Phi(t)^T Y(t)$ with $\Phi(t) := (\phi(0), \ldots, \phi(t))^T$ and $Y(t) := (Y(0), \ldots Y(t))^T$, this method of scaling can improve the numerical accuracy of the least squares calculations.

When scaling is used in recursive least squares estimation, the regressor vector is transformed by a *diagonal* scaling matrix before use in the estimator. The scaling matrix can be chosen to give minimum condition number for the covariance matrix P in some sense. Suppose that the estimate of $\hat{\theta}(t)$ using the unscaled (but normalized) regressor is given by equation (6) with $y(t)$ and $\phi(t)$ replaced by their normalized versions, $y^n(t)$ and $\phi_n(t)$ respectively. Now if $S(t)$ is the scaling matrix, chosen to minimize $C\{S(t)Q(t)\}$ (see Theorem 3 in Section 5.3) where $Q(t)Q(t)^T =: P(t)$, then the estimation with the scaled regressor, $\phi_{nS(t)}(t)$, is given by:

$$\hat{\theta}(t)=\hat{\theta}(t-1)+S(t)^{-1}\{P_{S(t)}(t)\phi_{nS(t)}(t)(y^{n}(t)-\hat{\theta}(t-1)^{T}\phi_{n}(t))\} \qquad (11)$$

where, $P_{S(t)}(t) := S(t)P(t)S(t)$ and $\phi_{nS(t)}(t) := S(t)^{-1}\phi_n(t)$. The scaling matrix is computed to minimize $C\{S(t)Q(t)\}$ rather than $C\{S(t)P(t)\}$ because of the need to preserve the symmetry of P after scaling.

In off-line or batch least squares estimation, the scaling matrix is a constant matrix and its value depends on the data. However, for recursive identification it is more logical to use a time-varying scaling matrix S(t), where S(t) is chosen at every time to minimize $C\{P_{S(t)}(t)\}$ as discussed above. However, if changing S(t) at every time t is not required or advisable (e.g. to reduce computational load) then S(t) may be changed whenever $C\{P_{S(t)}(t)\}$ exceeds a user specified bound κ.

The condition number, $C\{P_{S(t)}(t)\}$, is an important factor in ILS estimation. First, the asymptotic uncertainty in the parameter estimates (measured by their variance) depends on the measurement and structural (or modeling) errors and the condition number of P amplifies the effect of these errors. Niederlinski (1984) proves this result for the off-line or batch case. This result also holds true for the recursive case and is stated in the form of Theorem 2 in Section 5.3 and proven in Appendix 5.1 when the equation error term consists of white noise. Clearly it is desirable to have as small a condition number for P as possible. The data scaling and the on/off feature (discussed subsequently) ensure that this is so without altering the basic properties of P. An alternative is matrix regularization which ensures that the eigenvalues of P are bounded (equivalently, that the condition number of P is bounded) by modifying the P matrix. This may generate a different $\{\hat{\theta}(t)\}$ sequence, and perhaps parameter drifting when the system excitation is poor.

Secondly, the condition number of $P_{s(t)}(t)$ can also be used in lieu of the definition of persistence of excitation. A formal definition of persistency of excitation is given below in equation (13). If the condition number is large and application of the scaling matrix, S(t), does not significantly reduce $C\{P_{s(t)}(t)\}$ then it can be concluded that $P_{s(t)}(t)$ is almost singular and hence that the input data are not persistently exciting.

Thirdly, reducing the condition number of the covariance matrix improves the numerical performance of the ILS algorithm as discussed earlier.

The effect of data scaling is demonstrated by Examples 11 and 12 in Appendix 5.2. Compare these with Examples 6 and 7 where no scaling was employed. Scaling can improve the parameter estimates significantly.

Note that in the implementation of the proposed ILS algorithm and its testing the square-root of P(t) is computed using a Cholesky factorization. However, this factorization was found to be very sensitive to truncation errors. Therefore, as discussed earlier the recursive square-root approach used by standard RLS to calculate the square-root can also be used in ILS. This automatically gives a square-root of P(t) as well as ensuring $P(t) \geq 0, \forall t$. Using scaling further ensures that $C\{P_{s(t)}\}$ is minimized.

Step 5, Variable forgetting factor: The trace of P is a measure of the magnitude of P and hence of the magnitude of the estimator gain in equation (7). The covariance blowup experienced in least squares algorithms with a constant forgetting factor, $\lambda(t)=\lambda < 1$, may be eliminated by calculating a forgetting factor as given below to maintain a constant trace for P:

$$\lambda(t) = 1 - [r(t) - \{r(t)^2 - 4 \frac{\|P_{s(t)}(t-1)\phi_{ns(t)}(t)\|^2}{\operatorname{tr} P_{s(t)}(t-1)}\}^{1/2}]/2 \qquad (12)$$

where $r(t) := 1 + \phi_{ns(t)}(t)^T P_{s(t)}(t-1)\phi_{ns(t)}(t)$. This equation is easily derived from equation (8) (in which $\phi(t)$ and $P(\cdot)$ are replaced by $\phi_{ns(t)}(t)$ and $P_{s(t)}(\cdot)$ respectively) by taking the trace of both sides and setting $\operatorname{tr} P_{s(t)}(t) = \operatorname{tr} P_{s(t)}(t-1)$. A change in the specified value of tr P, e.g. to increase tr P and hence gain $\underline{K}(t)$ during periods of rich excitation, can be implemented by multiplying the right side of equation (8) by a scalar. The use of constant trace and the resulting improvements in the behaviour of the parameter estimates is illustrated by Examples 7, 9, 12, 14 and 15 in Appendix 5.2. For example, a comparison of the parameter estimates in Figure 5.26 versus 5.28 shows that specifying tr P=1 results in slower (smoother) parameter changes than tr P=4. The effect is similar to filtering but does not alter the least squares properties. It follows directly from equation (6).

Note that by calculating $\lambda(t)$ to keep $\operatorname{tr} P_{s(t)}(t)=\operatorname{tr} P_{s(t)}(t-1)$ one ensures that the trace of the original (i.e. unscaled) covariance matrix is kept constant. Thus the scaling does not affect the effective gain of the estimator. But whenever the scaling matrix is changed the trace of the scaled matrix changes (i.e. $\operatorname{tr} P_{s(t)}(t)\neq\operatorname{tr} P_{s(t-1)}(t)\neq\operatorname{tr} P(t)$).

Step 6, On/off criterion: One requirement of all estimators is that there must be sufficient excitation during periods of identification to excite all the relevant (or dominant) modes of the process. If there are relatively long periods during which the parameter estimation is active, but there is insufficient excitation, the estimator may give drifting parameter estimates due to noise etc. Many authors suggest the addition of "persistent excitation" during the estimation, but although this solves the problem, it is not always practical. An alternative approach is to stop estimation when the data are not persistently exciting.

Loosely speaking, a system is said to be persistently excited whenever there are a sufficient number of frequencies in the system input signal to excite all the relevant modes of the process and hence enable accurate estimation of all the desired plant parameters, rather than a restricted number of linear functionals of the parameters.

Mathematically, $\{\phi_{ns(t)}(t)\}$ is said to be persistently exciting (Astrom and Bohlin (1971)) whenever there exists a_1, a_2 >0 such that

$$a_1 I \ge \sum_{i=t-p+1}^{t} \phi_{nS(t)}(i)\phi_{nS(t)}(i)^T \ge a_2 I \qquad (13)$$

$\forall t$ and for some $p \ge \dim(\phi_{ns(t)}(\cdot))=n+m$.

A rigorous way of testing for persistency of excitation is to verify equation (13) at every time instant. Since this is too cumbersome numerically, measures such as $\|P_{S(t)}(t-1)\phi_{ns(t)}(t)\|$ or $C\{P_{S(t)}(t)\}$ may be used.

Step 6.1: The parameter update at each time interval takes place in the direction of the gain vector K(t) := $P_{s(t)}(t)\phi_{ns(t)}(t)$ as can be observed in equation (6) (with appropriate notational changes made due to normalization and scaling). $P_{s(t)}(t)$ is essentially a projection operator that projects the regressor vector in a direction useful for parameter updating. The P matrix enhances the directions in which little information is already at hand (corresponding to large eigenvalues of $P_{s(t)}(t)$) and reduces the influence of directions in which much information is already present (corresponding to small eigenvalues of $P_{s(t)}(t)$). This can be observed by taking $\phi_{ns(t)}(t)$ along the eigenvectors of $P_{s(t)}(t)$ i.e. by expanding $P_{s(t)}(t)\phi_{ns(t)}(t)$ as $\sum_{i=1}^{p} \mu_i\{P_{s(t)}(t)\}w_i w_i^T\phi_{ns(t)}(t)$ where $\mu_i\{P_{s(t)}(t)\}$ is the i^{th}-eigenvalue of $P_{s(t)}(t)$, w_i is the corresponding eigenvector and p=n+m. Thus $\|P_{s(t)}(t)\phi_{ns(t)}(t)\|$ (or equivalently $\|P(t-1)\phi_n(t)\|$) can be thought of as a rough measure of *new* information entering the estimation scheme and hence is an excellent basis for an on/off criterion. Therefore the estimation is stopped whenever $\|P(t-1)\phi_n(t)\| < \iota$ where ι is a user specified number and the algorithm jumps to step 6.5.

It may be noted in equation (13) that $\alpha_1 = \mu_{max}\{\sum_{i=t-p+1}^{t}\phi_{ns(t)}(t)\phi_{ns(t)}(t)^T\}$ and $\alpha_2 = \mu_{min}\{\sum_{i=t-p+1}^{t}\phi_{ns(t)}(t)\phi_{ns(t)}(t)^T\}$. Thus the ratio $\alpha_1/\alpha_2 =: C\{\sum_{i=t-p+1}^{t}\phi_{ns(t)}(t)\phi_{ns(t)}(t)^T\}$ may be used as a good measure of excitation. When the excitation is rich $\alpha_2 > 0$ and $C\{\sum_{i=t-p+1}^{t}\phi_{ns(t)}(t)\phi_{ns(t)}(t)^T\}$ will be well bounded. When the excitation is poor α_2 -> 0 and $C\{\sum_{i=t-p+1}^{t}\phi_{ns(t)}(t)\phi_{ns(t)}(t)^T\}$ may be arbitrarily large (or even unbounded). Since $P_{s(t)}(t)^{-1} \cong \sum_{i=t-p+1}^{t}\phi_{ns(t)}(t)\phi_{ns(t)}(t)^T$ when a forgetting factor is used, $C\{\sum_{i=t-p+1}^{t}\phi_{ns(t)}(t)\phi_{ns(t)}(t)^T\}$ may be approximated by $C\{P_{s(t)}(t)^{-1}\}$ (or equivalently $C\{P_{S(t)}(t)\}$). The estimation can be stopped if $C\{P_{S(t)}(t)\} > \kappa$ for some user specified upperbound κ. Note that large $C\{P_{S(t)}(t)\}$ may also occur due to poorly conditioned data (e.g. magnitudes of u and y are widely different). Hence appropriate scaling should be used before applying this test. Also note that it is relatively easy to compute an upper bound for $C\{P_{S(t)}(t)\}$, which is simply $\|P_{S(t)}(t)\|_\infty \ \|P_{S(t)}(t)^{-1}\|_\infty$ ($C\{P_{S(t)}(t)\} := \frac{\mu_{max}\{P_{S(t)}(t)\}}{\mu_{min}\{P_{S(t)}(t)\}} = \|P_{S(t)}(t)\|_2\|P_{S(t)}(t)^{-1}\|_2 \leq \|P_{S(t)}(t)\|_\infty\|P_{S(t)}(t)^{-1}\|_\infty$), because $\|\cdot\|_\infty$ of a matrix is easier to compute than a $\|\cdot\|_2$. $P_{S(t)}(t)^{-1}$ is updated recursively. The additional steps required to implement the ILS algorithm are as follows:

Step 6.2: Compute

$$P' = \left(I - \frac{P_{S(t-1)}(t-1)\phi_{nS(t-1)}(t)\phi_{nS(t-1)}(t)^T}{\lambda(t) + \phi_{nS(t-1)}(t)^T P_{S(t-1)}(t-1)\phi_{nS(t-1)}(t)}\right) \frac{P_{S(t-1)}(t-1)}{\lambda(t)}$$

$$P'^{-1} = \lambda(t)\ P_{S(t-1)}(t-1)^{-1} + \phi_{nS(t-1)}(t)\phi_{nS(t-1)}(t)^T$$

$$C\{P'\} = \|P'\|_\infty\ \|P'^{-1}\|_\infty$$

If $C\{P'\} > \kappa$ go to step 6.4

Step 6.3: Update $\hat{\theta}$, $P_{S(t-1)}$, $P_{S(t-1)}^{-1}$ and $S(t-1)$, i.e.

$P_{S(t)}(t) = P'$, $P_{S(t)}(t)^{-1} = P'^{-1}$, $S(t) = S(t-1)$,

$\hat{\theta}(t)=\hat{\theta}(t-1)+S(t)^{-1}P_{S(t)}(t)\phi_{nS(t)}(t)(y^n(t)-\hat{\theta}(t-1)^T\phi_n(t))$.

Go to step 7.

Step 6.4: Obtain the new scaling matrix S_{new} that minimizes $C\{S_{new}Q\}$ where Q is a Cholesky factor of P', i.e. $QQ^T =: P'$, and compute $P_{S(t)}' := S_{new}P'S_{new}$ and $P_{S(t)}'^{-1} := S_{new}^{-1}P'^{-1}S_{new}^{-1}$. If $C\{P_{S(t)}'\} < \kappa$ then go to step 6.6.

Step 6.5: Stop estimation, i.e.

$P_{S(t)}(t)=P_{S(t-1)}(t-1)$, $P_{S(t)}(t)^{-1}=P_{S(t-1)}(t-1)^{-1}$, $S(t)=S(t-1)$,

$\hat{\theta}(t)=\hat{\theta}(t-1)$. Return to step 1 at next sample instant.

Step 6.6: Update $\hat{\theta}$, $P_{S(t-1)}$, $P_{S(t-1)}^{-1}$, and $S(t-1)$, i.e.

$P_{S(t)}(t)=P_{S(t)}$, $P_{S(t)}(t)^{-1}=P_{S(t)}^{-1}$, $S(t)=S(t-1)S_{new}$,

$\phi_{nS(t)}(t)=S(t)^{-1}\phi_n(t)$

$\hat{\theta}(t)=\hat{\theta}(t-1) + S(t)^{-1}P_{S(t)}(t)\phi_{nS(t)}(t)(y^n(t) - \hat{\theta}(t-1)^T\phi_n(t))$

Example (14) in Appendix 5.2 demonstrates the use of the on/off criteria and shows that their use gives rise to smoother parameter estimates with less drift (e.g. compare Figure 5.17 versus 5.23).

Step 7, Use of prior knowledge:It seems intuitively plausible that the estimation scheme will be more robust and efficient if prior knowledge about the model structure or model parameters can be employed. For example, if prior information specifying a known region in which the parameters lie is available, then the parameter estimation procedure may be improved by projecting them to lie inside this region. Usually knowledge regarding the dominant time constant, damping factor, steady state gain etc. of the continuous plant are known and it is not too difficult to map this information into knowledge regarding the discrete time model parameters being estimated. In implicit adaptive control applications the estimated controller parameters could be constrained (or projected) into an *a priori* known stable region.

Several ways of projecting (or constraining) the parameter estimates have been suggested (e.g. Goodwin and Sin (1984), Ossman and Kamen (1986), Praly (1983)) but the simplest one (due to Praly) is based on the idea of projecting the parameters into a sphere of known centre, θ_c, and radius, R:

$$\hat{\theta}_p(t) = \theta_c + \min(1, R/\|\theta_D(t)\|)\ \theta_D(t) \qquad (14)$$

where $\theta_D(t) := \hat{\theta}(t) - \theta_c$. Thus if the new estimate $\hat{\theta}(t)$ is outside the sphere it is replaced by its projection $\hat{\theta}_p(t)$. One feature of this projection scheme is that it alters the convergence properties of the estimator. Parameter projection is particularly useful when large, unexpected load type disturbances frequently upset the plant and cause parameter estimates to move significantly. After parameter projection the ILS algorithm returns to step 1 at the next sampling instant.

Other types of prior information that can profitably be used to improve the parameter identification include known non-linearities (e.g. a known non-linear static gain as in a Hammerstein model, Anbumani *et al.* (1981)) and known system poles and/or zeros (Bai and Sastry (1986)).

5.3 Convergence behaviour of the improved least squares estimator

Theoretical convergence analysis of an estimator provides insight into how the parameter estimates behave and is an important step in the proof of stability of an adaptive control scheme. Two theorems are presented in this section that summarize the convergence behaviour of the proposed ILS algorithm in the deterministic and stochastic cases assuming that the plant is time-invariant and that upperbounds n and m on the orders of the A and B polynomials in the plant model are known. In the stochastic case, the condition number of the P matrix has a significant effect on the parameter estimates and the theorem brings out this fact. A third theorem specifies how the scaling matrix, S(t), discussed in section 5.2.3 can be computed.

Theorem 1: The constant trace least squares algorithm given by equations (6), (7) and (8) with $y(t)$ and $\phi(t)$ replaced by their appropriately normalized versions $y^n(t)$ and $\phi_n(t)$ respectively and with $\lambda(t)$ as given by equation (13) applied to the deterministic (perturbation-free) plant

$$y^n(t)=\phi_n(t)^T\theta_* \tag{15}$$

where θ_* is a constant vector, has the following properties:

1. $P(t)^{-1} = \lambda(t)\, P(t-1)^{-1} + \phi_n(t)\phi_n(t)^T$
2. $\theta(t) = \theta(t-1) - P(t)\phi_n(t)e(t)$
   where $\theta(t) := \theta_* - \hat{\theta}(t)$
   and $e(t) := y^n(t) - \phi_n(t)^T\hat{\theta}(t-1)$
3. $P(t)^{-1}\theta(t) = \lambda(t)P(t-1)^{-1}\theta(t-1)$
4. $0<\lambda(t)\leq 1$
5. $V(t) = \lambda(t)V(t-1) - \dfrac{\lambda(t)e(t)^2}{\lambda(t) + \phi_n(t)^TP(t-1)\phi_n(t)}$
   where $V(t) := \theta(t)^TP(t)^{-1}\theta(t)$ is a Lyapunov function
6. $\|\theta(t)\|^2 \leq \kappa(t-1)\|\theta(t-1)\|^2$
   $\|\theta(t)\|^2 \leq \kappa(0)\|\theta(0)\|^2$
   where $\kappa(t) = C\{P(t)\} := \dfrac{\mu_{max}\{P(t)\}}{\mu_{min}\{P(t)\}}$ and $\mu_{max}\{P(t)\}$ ($\mu_{min}\{P(t)\}$) is the maximum (minimum) eigenvalue of $P(t)$
7. $\lim_{t\to\infty} (\hat{\theta}(t) - \hat{\theta}(t-k)) = 0$ for any finite integer k
8. $\lim_{t\to\infty} e(t) = 0$
   Note that properties (8) and (9) imply asymptotic parameter convergence to constant values such that $\hat{y}(t) := \hat{\theta}(t)^T\phi_n(t) \to y^n(t)$

9. Let $\delta := \lim_{t\to\infty} [\prod_{i=1}^{t} \lambda(i)]$, $0 \le \delta \le 1$

If $\delta = 0$ then $\lim_{t\to\infty} \hat{\theta}(t) = \theta_*$

If $\delta \ne 0$ then as $t \to \infty$

(i) $\lambda(t) \to 1$

(ii) $\|P(t-1)\phi_n(t)\| \to 0$

(iii) $P(t) \to P_\infty$

(iv) $\hat{\theta}(t) \to \hat{\theta}_\infty$

Proof: (see Appendix 5.1)

Note: Property (9) is due to Lozano and Goodwin (1985).

Remark 1: Note that data preprocessing operations such as d.c. value elimination and scaling do not change the results of the above theorem as long as the true parameter vector θ_* is interpreted appropriately. The effect of modeling errors is not considered in this analysis.

Remark 2: When the parameter projection scheme outlined in section 5.2.3 is used $V(t)$ ceases to be a Lyapunov function and properties (7) and (8) do not directly follow. However, assuming that the true parameter vector, θ_*, lies inside the sphere it can be shown that the convergence property (6) still holds.

Remark 3: The least squares algorithm has exponential convergence (Johnstone *et al.* (1982)) when the forgetting factor $\lambda(t)=\lambda < 1$ and the data are persistently exciting, in the sense that there exists a constant $A > 0$ such that

$$\|\theta(t)\|^2 \le A\lambda^t. \tag{16}$$

The equivalent result in the variable forgetting factor case is stated by the property (9). In this case it can be shown that

$$\|\theta(t)\|^2 \le A[\prod_{i=1}^{t} \lambda(i)] \tag{17}$$

Since $\lambda(t)$ is variable an exponential convergence result similar to equation (16) can only be specified in terms of an upper bound, λ', on the sequence $\{\lambda(t)\}$ as

$$\|\theta(t)\|^2 \le A\lambda'^t \tag{18}$$

If the data are persistently exciting then $\|P(t-1)\phi_n(t)\| > 0$ and $\lambda(t) < 1 \; \forall t$ and hence there exists $\lambda' < 1$ such that the convergence rate is exponential in the variable forgetting factor case.

Remark 4: Most identification algorithms exhibit exponential convergence to the true parameter values under conditions of rich excitation. The least squares algorithm which is essentially a Newton algorithm, uses the curvature information of the sum of squares error surface and therefore exhibits faster (quadratic) convergence, particularly when the estimated parameters are close to their true values (i.e. the covariance matrix can therefore be thought of as including including second order or curvature information). The proposed ILS algorithm preserves this property as it uses a true (weighted) least squares step at each parameter update which is an important feature in practical applications.

Theorem 2: The improved least squares algorithm with normalization and constant trace features has the following properties when applied to the system:

$$y^n(t) = \phi_n(t)^T\theta_* + \eta^n(t) \qquad (19)$$

where θ_* is a constant vector and $\{\eta^n(t)\}$ is a white noise sequence of variance γ^2 and mean zero:

1. $\mu_{min}\{P(t)^{-1}\} > p_{min} \geq 0$
2. $\mu_{max}\{P(t)^{-1}\} \leq p_{max} = 1/\epsilon$
3. $E\{\theta(t)/F(t-1)\} = 0$ a.s.
4. $E\{\|\theta(t)\|^2/F(t-1)\} \leq M\gamma^2 = \dfrac{p_{max}}{p_{min}}\gamma^2$ a.s.

where $F(t-1)$ are the increasing subsigma algebras generated by the plant inputs and outputs available up to time (t-1), M is an upper bound on the condition number of $P(t)^{-1}$ (or equivalently $P(t)$). If δ and ϵ are constants such that $0 < \delta \leq \lambda(t) \leq 1 - \epsilon \; \forall t \; (\epsilon \geq 0)$ then $p_{min} = a_2\delta^p$, $t \geq p$ where a_2 is the lower bound in equation (13).

Proof: (see Appendix 5.1)

Note: Lozano (1983) obtains a similar result by assuming that $\{\phi_n(t)\}$ is persistently exciting. This assumption is not needed in the present case.

Remark 5: If data are persistently exciting then $a_2 > 0$ and $\epsilon > 0$ and hence $M < \infty$. The on/off criterion makes sure that there is enough excitation and hence M is well bounded during the periods in which estimation is active. In the proposed ILS algorithm $\|P(t-1)\phi_n(t)\|$ and the condition number of P after minimization are used as the basis for the on/off decision.

Remark 6: The trace of the covariance matrix also has an effect on the variance of the parameter estimates. It can be shown that $E\{\theta(t)\theta(t)^T\} \cong P(t)\gamma^2$ and therefore $E\{\|\theta(t)\|^2\} \cong$ tr $P(t)\gamma^2$ which means that a larger trace for P also leads to a larger variance of the estimates. Therefore the selection of the user specified value for tr P in ILS is a trade-off between increasing the tracking ability (i.e. gain in equation (6)) and reducing the variance of the estimated parameters. For example, when good excitation such as a setpoint change is expected a larger trace for P may be used. When parameter estimates seem to have converged or when the process is at steady state a smaller trace for P may be used. Note that a small trace for P results in smaller variations in $\hat{\theta}(\cdot)$ and therefore has a filtering effect on $\hat{\theta}(\cdot)$. However, unlike adhoc filtering schemes for $\hat{\theta}(\cdot)$, specification of a small tr P preserves the properties of the least squares estimation.

Theorem 3 (Computing the scaling matrix S): A diagonal S that minimizes $C\{SA\} := \|SA\|_\infty \|A^{-1}S^{-1}\|_\infty$ is given by choosing $s_{ii}=1/r_i$ where r_i is the i^{th} absolute row-sum (i.e. sum of the absolute values of the elements of the i^{th} row) of A. The row-sums of SA are then all equal.

Proof: (See Noble 1969)

5.4 Literature review

In this section solutions proposed in the literature to some of the problems in least squares parameter estimation are briefly reviewed and compared with the features of the ILS algorithm presented in this chapter.

5.4.1 Non-zero mean data

The d.c. bias in the process, which typically leads to parameter offset if not properly taken into account, has received a lot of attention in the literature. The bias is typically generated by non-zero mean disturbances or when a linear model is fitted to a non-linear process around a series of new operating points having different non-zero mean values (for U and Y). The method suggested in this chapter to solve the bias problem involves estimating the d.c. values $U_{d.c.}$ and $Y_{d.c.}$ of the input/output signals and using the zero mean deviations $u(\cdot)$ and $y(\cdot)$ from these estimated d.c. values in the parameter estimation. This method has the advantage that the parameter estimation can be frozen without adversely affecting the tracking of the d.c. bias. Other methods suggested in the literature include estimating a bias term as a part of the parameter vector (which is called a "1-in-the-regressor-vector" method) and estimation based on incremental (or differenced) data (Tuffs and Clarke (1985)).

There are several problems associated with the "1-in-the-regressor-vector" method. First, the method works well only for a constant or slowly time-varying d.c. bias. Second, if parameter estimation is frozen then the d.c. level estimation is also frozen, leading to prediction offsets. Third, it is not straightforward to constrain the parameter estimates to known regions with this method.

In the incremental estimation approach, the input/output data are first differenced before the dynamic plant parameters are estimated. An advantage of estimating on the basis of differenced data is that there is no need to explicitly include a d.c. level in the plant model. A disadvantage is that the effects of high-frequency noise are amplified because of the differencing (the variance of the differenced data is twice the variance of the original data).

The method proposed in this chapter works well when the d.c. bias is constant and high-frequency noise is significant. By a careful choice of the forgetting factors $\lambda_y(t)$ and $\lambda_u(t)$ in the d.c. value estimation step the proposed scheme may be used even when the d.c. bias is slowly varying. Note that the approach recommended here is designed to eliminate the effect of a constant (or steady state) bias on the estimated parameters and does not eliminate the effect of dynamic disturbances. An extended least squares approach using the proposed ILS algorithm could be used to deal with dynamic disturbances.

5.4.2 Model Process Mismatch

Although model process mismatch, MPM, may exist for many reasons, the most commonly stated cause is that due to reduced-order modeling (ROM). Several solutions were proposed to handle MPM based on stability as well as heuristic considerations. These include: (1) normalization of the regressor vector plus the use of a dead-zone in the estimator (Cluett *et al.* (1986)), (2) regressor filtering (Astrom (1983)), (3) use of a smaller estimator gain (Rohrs *et al.* (1984)), (4) use of a persistently exciting input signal with appropriate frequency content (Wittenmark and Astrom (1984)) and (5) the method of averaging and slow sampling (Rohrs *et al.* 1984; Astrom (1983)).

A dead-zone was also suggested for handling bounded noise (Goodwin and Sin (1984) and Martin-Sanchez *et al.* (1982)). The idea of a dead-zone where parameter estimation is switched off is basically motivated by stability considerations and is not intuitively appealing, because the estimation may be stopped even when continued estimation would improve the parameter estimates.

Regressor filtering is a heuristic fix that has been found to work well in practice. However, there are few guidelines for the design of the filter. Filtering can also be used to give the signals appropriate frequency content.

The addition of persistently exciting signals ensures that the estimation scheme is exponentially convergent which gives a measure of robustness to the adaptive system in the presence of small amounts of measurement and modeling errors.

It has been shown by Rohrs *et al.* (1985) that the unmodeled (high frequency) effects can be removed by sampling the system slowly enough. The method of averaging which is based on the idea of estimating only once in a few sampling intervals instead of at every sampling interval has essentially the same effect as slow sampling.

The proposed ILS algorithm accommodates most of the above mentioned methods. For example, it is active only during periods of persistent excitation, includes normalization plus a means for user specification of the estimator gain (via tr P), and allows the use of filtering, the addition of excitation and averaging or slow sampling.

5.4.3 Tracking time varying parameters

Tracking of time-varying parameters requires a non-zero gain in the estimator. Several modifications have been proposed in the literature to the least squares estimator to prevent turn-off. Most of the modifications are adhoc and create other problems, e.g. the variable forgetting factor approach due to Fortescue *et al.*(1981) may fail in noisy systems. The variable forgetting factor approach proposed in this chapter to obtain a constant trace for P preserves the weighted least squares direction at each parameter update. The idea of varying λ to control the trace of P was first suggested by Irving (1981). Lozano and Goodwin (1985) used the same constant trace approach with a least squares like identification scheme.

Goodwin *et al.* (1984) suggest periodic or regular resetting of the covariance matrix to maintain estimator sensitivity to process parameter variations. In their latest modification (Goodwin *et al.* (1985)) they reset P regularly to maintain its trace at a constant value. However, resetting the P matrix alters the least squares algorithm since the directional information stored in the P matrix is lost. Therefore its convergence behaviour, loosely speaking, will be somewhere between the projection and true least squares identification schemes. In stochastic simulation studies it has been observed that convergence difficulties sometimes arise when P is changed significantly during the covariance resetting step. However, analysis of the algorithm has so far failed to define the cause or conditions that lead to this problem. One advantage of covariance resetting is that the condition number of P is automatically bounded. However, with resetting the C(P) can no longer be used as a measure of persistent excitation as suggested herein. It appears that reducing C(P) through scaling and bounding it through on/off gives significantly better results than covariance resetting.

Some comparisons of the proposed constant trace approach with that of Goodwin's method are presented in Example 15 of Appendix 5.2. This example also illustrates the effect of the size of tr P on the estimated parameters.

5.5 Conclusions

Improved numerical conditioning: The numerical properties of the algorithm are improved by scaling each element of the regressor vector, $\phi(t)$, by the norm of $\phi(t)$. Also, $\phi_n(t)$ is scaled by a diagonal scaling matrix, S(t), selected to minimize $C\{P_{S(t)}(t)\}$ where $P_{S(t)}(t) = S(t)P(t)S(t)$. Simulations show that the effect of the data prescaling can be very significant. It has not been recommended previously for adaptive control applications. This scaling is applicable to other estimation schemes and to techniques such as Kalman filtering.

Constant tr P by varying the forgetting factor: This prevents covariance blowup which in turn could lead to "bursting" of U(t) and Y(t) and also maintains the tracking ability of the algorithm at a user specified value. The formula suggested for computing the forgetting factor $\lambda(t)$ is obtained from an exact weighted least squares estimator (i.e. no simplifications are used) and has not been suggested previously in the literature. This variable forgetting factor approach is superior to covariance resetting techniques (e.g. Goodwin *et al*. 1985) which change the parameter updating direction from that of the true least squares algorithm and make the algorithm more noise sensitive. Since the covariance matrix, P, (or P_s) is a measure of the error in the parameter estimates it is obviously desirable to keep P as small as possible while still permitting satisfactory tracking. Trace P can be varied on-line by multiplying P(t-1) by a suitable scalar. This "gain scheduling" approach has proven effective in applications, e.g. to speed up parameter adaptation during the intial startup phase and/or immediately following a setpoint change.

On/off criteria: Continued estimation in the absence of persistent excitation can lead to significant parameter drift. An on/off criterion can prevent this problem. However, if the on/off criteria are not chosen properly their use could lead to slower parameter convergence, e.g. if the algorithm were turned off while there was still significant new information present in the plant data. It is shown that $\|P_{s(t)}(t)\phi_{ns(t)}(t)\|$ is a measure of is the amount of useful new information in the regressor vector $\phi_{ns(t)}(t)$ and hence is an appropriate on/off criterion. Similarly, the condition number $C\{P_{s(t)}(t)\}$ is shown to be a measure of excitation. Since numerical effects due to $C\{P_{s(t)}(t)\}$ were minimized by the prescaling step, $C\{P_{s(t)}(t)\}$ can be used directly as a measure of excitation.

Independent estimation of the mean (or d.c.) values of U and Y: The effect of d.c. level in the U and Y elements of the regressor can be eliminated asymptotically by independently estimating the mean values $U_{d.c.}$ and $Y_{d.c.}$ and subtracting them from the new data. Least squares techniques (with variable forgetting factors) can be used to estimate the means. This approach is an alternative to methods such as "one-in-the-regressor-vector" method or incremental identification. When the forgetting factors are equal and constant it can be shown to be equal to filtering the regressor with $T(q^{-1})$.

Each of the above features improves the performance of the ILS algorithm under specific conditions. However, the key contributions are the selection and integration of all features into a single ILS algorithm plus the fact that the main theoretical properties of least squares estimation are retained.

The ultimate worth of any estimation algorithm is determined by its performance on practical applications. Unfortunately the performance of any estimation scheme is application dependent and hence it is difficult to *prove* the superiority of the proposed ILS algorithm by simply presenting a few examples. However, several simulated examples are presented in Appendix 5.2 which provide a good insight into the various problems associated with ordinary RLS and how ILS deals with these problems.

References

Anbumani, K., L., L. M. Patnaik, I. G. Sarma (1981): "Self-tuning Minimum Variance Control of Non-linear Systems of the Hammerstein Model", IEEE Trans. Aut. Control, AC-26, 959.

Anderson, B. D. O. (1985): "Adaptive Systems, Lack of Persistency of Excitation and Bursting Phenomena", Automatica, 21, 247.

Astrom, K. J. (1983): "Analysis of Rohrs' Counter Examples to Adaptive Control", Proc. of 22nd CDC, San Antonio, TX, USA, 713.

Astrom, K. J. and T. Bohlin (1965): "Numerical Identification of Linear Dynamic Systems from Normal Operating Records", Proceedings of the IFAC Symposium on Self-Adaptive Systems, Teddington, England, 96.

Astrom, K. J. and P. Eykhoff (1971): "System Identification - A Survey", Automatica, 7, 123.

Bai, Er Wei and S. Sastry (1986): "Adaptive Control of Partially Known Systems", Proc. of ACC, Seattle, WA, USA, 103.

Cluett, W. R., S. L. Shah and D. G. Fisher (1986): "Robust Adaptive Control in the Presence of Unmodeled Dynamics", Proc. of ACC, Seattle, WA, USA, 713.

Fortescue, T. R., L. S. Kershenbaum and B. E. Ydstie (1981): "Implementation of Self-tuning Regulators with Variable Forgetting Factors", Automatica, 17, 831.

Gilchrist, W. (1977): Statistical Forecasting, Wiley-Interscience.

Goodwin, G. C., D. J. Hill and M. Palaniswami (1985): "Towards an Adaptive Robust Controller", Proc. of the 7th IFAC Symposium on Identification and System Parameter Estimation, York, UK, 997.

Goodwin, G. C. and K. S. Sin (1984): Adaptive Filtering, Prediction and Control, Prentice-Hall.

Irving, E. and H. Dang Van Mien (1981): "Discrete-time Model Reference Multivariable Adaptive Control: Applications to Electrical Power Plants", Proc. of IFAC 8th Triennial World Congress, Kyoto, Japan, 227.

Iserman, R. (1980): "Paramter Adaptive Control Algorithms", Automatica, 16, 513.

Johnstone, R. M., C. R. Johnson (Jr.), R. R. Bitmead and B. D. O. Anderson (1982): "Exponential Convergence of Recursive Least Squares with Forgetting Factor", Systems and Control Letters, 2, 77.

Lozano-Leal, R. (1983): "Convergence Analysis of Recursive Identification Algorithms with Forgetting Factor", Automatica, 19, 95.

Lozano-Leal, R. and Goodwin, G. C. (1985): "A Globally Convergent Adaptive Pole Placement Algorithm without a Persistency of Excitation Requirement", IEEE Trans. Aut. Control, AC-30, 795.

Ljung, L. and T. Soderstrom (1983): Theory and Practice of Recursive Identification, The MIT Press.

Martin-Sanchez, J. M., S. L. Shah and D. G. Fisher (1982): "A Stable Adaptive Predictive Control System", Intl. J. Control, 39, 215.

Niederlinski, A. (1984): "A New Look at Least Squares Dynamic System Identification", Intl. J. Control, 40, 467.

Nieman, R. E. (1971): Ph. D. Thesis, Department of Chemical Engineering, University of Alberta, Edmonton, Canada.

Noble, B. (1969): Applied Linear Algebra, Prentice-Hall.

Ossman, K. and E. W. Kamen (1986): "A Parameter Estimator with Convergence to Pre-specified Intervals for Discrete Time Systems", Proc. of ACC, Seattle, WA, USA, 1073.

Praly, L. (1983): "Robustness of Model Reference Adaptive Control", Proc. of the 3^{rd} Yale Workshop on Applications of Adaptive Systems Theory, 253.

Rohrs, C. E., M. Athans, L. S. Valavani and G. Stein (1984): "Some Design Guidelines for Discrete-time Adaptive Controllers", Automatica, 20, 653.

Rohrs, C. E., G. Stein and K. J. Astrom (1985): "A practical robustness theorem for adaptive control", Proc. of ACC, 979.

Sinha, N. K. and B. Kuszta (1983): Modeling and Identification of Dynamic Systems (Van Nostrand Reinholdt).

Strejc, V. (1980): "Least Squares Parameter Estimation", Automatica, 16, 535.

Strejc, V. (1981): "Trends in Identification", Automatica, 17, 7.

Tuffs, P. S. and D. W. Clarke (1985): "Self-tuning Control Of offset: A Unified Approach", IEE Proc., 132, D, 104.

Young, P. (1984): Recursive Estimation and Time-series Analysis - An Introduction, Springer-Verlag.

Wittenmark, B. and K. J. Astrom (1984): "Practical Issues in the Implementation of Self-tuning Controllers", Automatica, 20, 595.

Woodall, W. H. (1986): "The Design of CUSUM Quality Control Charts", Journal of Quality Technology, 18, 2, 99.

Appendix 5.1

Proof of theorem 1 (Note that some of the steps are available in the literature but are included here for continuity).

part (i).

Since $P(t)^{-1} = \sum_{i=0}^{t} [\prod_{j=i+1}^{t} \lambda(j)]\phi_n(i)\phi_n(i)^T$

$$= \sum_{i=0}^{t-1} [\prod_{j=i+1}^{t} \lambda(j)]\phi_n(i)\phi_n(i)^T + \phi_n(t)\phi_n(t)^T$$

$$= \lambda(t)\sum_{i=0}^{t-1} [\prod_{j=i+1}^{t-1} \lambda(j)]\phi_n(i)\phi_n(i)^T + \phi_n(t)\phi_n(t)^T$$

$$= \lambda(t)P(t-1)^{-1} + \phi_n(t)\phi_n(t)^T$$

part (ii). Follows directly by subtracting equation (6) from θ_*.

part (iii).

From part (ii), noting that $e(t) = \phi_n(t)^T\theta(t-1)$, we have

$$\theta(t) = \theta(t-1) - P(t)\phi_n(t)\phi_n(t)^T\theta(t-1)$$

$$= (I - P(t)\phi_n(t)\phi_n(t)^T)\theta(t-1)$$

Multiplying both sides by $P(t)^{-1}$, we get

$$P(t)^{-1}\theta(t) = (P(t)^{-1} - \phi_n(t)\phi_n(t)^T)\theta(t-1)$$

$= \lambda(t)P(t-1)^{-1}\theta(t-1)$ from part (i).

part (iv). Proof is heuristic.

part (v).

$$\theta(t)^TP(t)^{-1}\theta(t) = (\theta(t-1) - P(t)\phi_n(t)e(t))^TP(t)^{-1}(\theta(t-1) - P(t)\phi_n(t)e(t))$$

$$= \theta(t-1)^TP(t)^{-1}\theta(t-1) - 2e(t)^2 + \phi_n(t)^TP(t)\phi_n(t)e(t)^2$$

$$= \theta(t-1)^T(\lambda(t)P(t-1)^{-1} + \phi_n(t)\phi_n(t)^T)\theta(t-1) - 2e(t)^2 + \frac{\phi_n(t)^TP(t-1)\phi_n(t)}{\lambda(t) + \phi_n(t)^TP(t-1)\phi_n(t)}e(t)^2$$

$$=\lambda(t)\theta(t-1)^T P(t-1)^{-1}\theta(t-1) + (\theta(t-1)^T\phi_n(t))^2 -$$

$$2e(t)^2$$

$$+ \frac{\phi_n(t)^T P(t-1)\phi_n(t)}{\lambda(t) + \phi_n(t)^T P(t-1)\phi_n(t)} e(t)^2$$

$$=\lambda(t)\theta(t-1)^T P(t-1)^{-1}\theta(t-1)$$

$$- \frac{\lambda(t)e(t)^2}{\lambda(t) + \phi_n(t)^T P(t-1)\phi_n(t)}$$

part (vi).

From part (v) $\theta(t)^T P(t)^{-1}\theta(t) \le \lambda(t)\theta(t-1)^T P(t-1)^{-1}\theta(t-1)$

$\Rightarrow \mu_{min}\{P(t)^{-1}\} \, \|\theta(t)\|^2 \le \lambda(t)\mu_{max}\{P(t-1)^{-1}\} \, \|\theta(t-1)\|^2$

But from part (i) $\mu_{min}\{P(t)^{-1}\} \ge \lambda(t) \, \mu_{min}\{P(t-1)^{-1}\}$

$\therefore \mu_{min}\{P(t-1)^{-1}\} \, \|\theta(t)\|^2 \le \lambda(t)\mu_{max}\{P(t-1)^{-1}\} \, \|\theta(t-1)\|^2$

$\Rightarrow \|\theta(t)\|^2 \le \kappa(t-1) \, \|\theta(t-1)\|^2$

where $\kappa(t-1) := \dfrac{\mu_{max}\{P(t-1)^{-1}\}}{\mu_{min}\{P(t-1)^{-1}\}}$

$$\theta(t)^T P(t)^{-1}\theta(t) \le \lambda(t) \, \theta(t-1)^T P(t-1)^{-1}\theta(t-1)$$

$$\le\lambda(t)\lambda(t-1)\theta(t-2)^T P(t-2)^{-1}\theta(t-2)$$

$$\ldots$$

$$\le\Big[\prod_{i=1}^{t} \lambda(i)\Big] \, \theta(0)^T P(0)^{-1}\theta(0)$$

Similarly from part (i)

$$\mu_{min}\{P(t)^{-1}\} \ge \lambda(t) \, \mu_{min}\{P(t-1)^{-1}\}$$

$$\ldots$$

$$\ge \Big[\prod_{i=1}^{t} \lambda(i)\Big] \, \mu_{min}\{P(0)^{-1}\}$$

$$\therefore \Big[\prod_{i=1}^{t} \lambda(i)\Big] \, \mu_{min}\{P(0)^{-1}\} \, \|\theta(t)\|^2 \le \Big[\prod_{i=1}^{t} \lambda(i)\Big] \, \mu_{max}\{P(0)^{-1}\} \, \|\theta(0)\|^2$$

$\Rightarrow \|\theta(t)\|^2 \le \kappa(0) \, \|\theta(0)\|^2$

part (viii).

From part (v),

$$V(t) = \lambda(t) \, V(t-1) - \frac{\lambda(t)e(t)^2}{\lambda(t) + \phi_n(t)^T P(t-1)\phi_n(t)}$$

$$\ldots$$

$$=[\prod_{i=1}^{t} \lambda(i)]\, V(0) - \sum_{i=1}^{t} [\prod_{j=i}^{t} \lambda(j)] \frac{e(i)^2}{\lambda(i) + \phi_n(i)^T P(i-1)\phi_n(i)}$$

Since $V(t) \geq 0$,

$$\sum_{i=1}^{t} [\prod_{j=i}^{t} \lambda(j)] \frac{e(i)^2}{\lambda(i) + \phi_n(i)^T P(i-1)\phi_n(i)} \leq [\prod_{i=1}^{t} \lambda(i)] V(0)$$

Since $V(0)$ is bounded, taking limits as $t \to \infty$ and using result (9) (which is proved later) we can conclude that

$$\lim_{t\to\infty} e(t) = 0$$

part (vii).

From (viii) we can say that

$$\lim_{t\to\infty} \sum_{i=1}^{t} \frac{e(i)^2}{\lambda(i) + \phi_n(i)^T P(i-1)\phi_n(i)} < \infty \qquad \text{(A.1)}$$

$$\Rightarrow \lim_{t\to\infty} \sum_{i=1}^{t} \frac{[\lambda(i) + \phi_n(i)^T P(i-1)\phi_n(i)]e(i)^2}{[\lambda(i) + \phi_n(i)^T P(i-1)\phi_n(i)]^2} -< \infty \qquad \text{(A.2)}$$

$$\hat{\theta}(t) - \hat{\theta}(t-1) = \frac{P(t-1)\phi_n(t)}{\lambda(t) + \phi_n(t)^T P(t-1)\phi_n(t)} e(t)$$

$$\therefore \|\hat{\theta}(t) - \hat{\theta}(t-1)\|^2 = \frac{\phi_n(t)^T P(t-1) P(t-1)\phi_n(t)}{[\lambda(t) + \phi_n(t)^T P(t-1)\phi_n(t)]^2} e(t)^2$$

$$\leq \frac{\phi_n(t)^T P(t-1)\phi_n(t) e(t)^2}{[\lambda(t) + \phi_n(t)^T P(t-1)\phi_n(t)]^2} \mu_{max}\{P(t-1)\}$$

$$\leq \frac{\phi_n(t)^T P(t-1)\phi_n(t) e(t)^2}{[\lambda(t) + \phi_n(t)^T P(t-1)\phi_n(t)]^2} \operatorname{tr} P(t-1) \qquad \text{(A.3)}$$

$$\therefore \sum_{i=1}^{t} \|\hat{\theta}(i) - \hat{\theta}(i-1)\|^2 = \sum_{i=1}^{t} \frac{\phi_n(i)^T P(i-1)\phi_n(i) e(i)^2 \operatorname{tr} P(t-1)}{[\lambda(i) + \phi_n(i)^T P(i-1)\phi_n(i)]^2} \qquad \text{(A.4)}$$

Since $\operatorname{tr} P(t) = \text{constant } \forall t$, let $\operatorname{tr} P(t) = \operatorname{tr}$. Then from (A.1), (A.2) and (A.4) it follows that

$$\lim_{t\to\infty} \sum_{i=1}^{t} \|\hat{\theta}(i) - \hat{\theta}(i-1)\|^2 = \lim_{t\to\infty} \sum_{i=1}^{t} \frac{\phi_n(i)^T P(i-1)\phi_n(i)e(i)^2 tr}{[\lambda(i) + \phi_n(i)^T P(i-1)\phi_n(i)]^2}$$

$$< \infty \qquad \text{(A.5)}$$

$$\Rightarrow \lim_{t\to\infty} (\hat{\theta}(t) - \hat{\theta}(t-1)) = 0$$

The result stated in the theorem can now be proved as follows:

$$\|\hat{\theta}(t) - \hat{\theta}(t-k)\|^2 = \|(\hat{\theta}(t)-\hat{\theta}(t-1)) + (\hat{\theta}(t-1)-\hat{\theta}(t-2)) + \ldots + (\hat{\theta}(t-k+1)-\hat{\theta}(t-k))\|^2$$

$$\leq \|\hat{\theta}(t)-\hat{\theta}(t-1)\|^2 + \ldots + \|\hat{\theta}(t-k+1)-\hat{\theta}(t-k)\|^2 \qquad \text{(A.6)}$$

from the Schwarz inequality

$$\therefore \sum_{i=1}^{t} \|\hat{\theta}(i) - \hat{\theta}(i-k)\|^2 \leq \sum_{i=1}^{t} \|\hat{\theta}(i) - \hat{\theta}(i-1)\|^2 + \ldots + \sum_{i=1}^{t} \|\hat{\theta}(i-k+1) - \hat{\theta}(i-k)\|^2$$

Taking limits as $t \to \infty$ and noting (A.5) we get

$$\lim_{t\to\infty} \sum_{i=1}^{t} \|\hat{\theta}(i) - \hat{\theta}(i-k)\|^2 < \infty$$

which implies $\lim_{t\to\infty} (\hat{\theta}(t) - \hat{\theta}(t-k)) = 0$

part (ix).

From part (v) we note that

$$V(t) \leq \lambda(t)V(t-1) \leq \ldots \leq [\prod_{i=1}^{t} \lambda(i)]V(0)$$

$$\lim_{t\to\infty} V(t) = V_\infty \leq \delta V(0)$$

If $\lim_{t\to\infty} \theta(t) = \theta_\infty$ and $\lim_{t\to\infty} P(t) = P_\infty$ then

$$0 \leq \theta_\infty^T P_\infty^{-1} \theta_\infty \leq \delta V(0)$$

If $\delta=0$ then $\theta_\infty^T P_\infty^{-1}\theta_\infty=0$.

Since in the proposed algorithm tr $P(t)$ is maintained at a constant, bounded value $P_\infty^{-1}>0$ so that $\theta_\infty=0$ (or equivalently $\lim_{t\to\infty} \hat{\theta}(t) = \theta_*$)

If $\delta\neq0$ then

$\lambda(t) \to 1$ as $t \to \infty$, otherwise $\delta=0$.

If $\lambda(t) \to 1$ then $\|P(t-1)\phi_n(t)\| \to 0$.

Since $\|P(t-1)\phi_n(t)\| \to 0$ (i.e. estimation stops), $\hat{\theta}(t) \to \hat{\theta}_\infty$ as $t \to \infty$ ($\hat{\theta}_\infty$ may or may not be equal to θ_*)

Also $P(t) \to P_\infty$ as $t \to \infty$.

Proof of theorem 2.

part (i).

We know that $P(t)^{-1} = \lambda(t)P(t-1)^{-1} + \phi_n(t)\phi_n(t)^T$

$$\Rightarrow P(t)^{-1} = [\prod_{i=1}^{t} \lambda(i)]P(0)^{-1} + \sum_{i=1}^{t} [\prod_{j=i+1}^{t} \lambda(j)]\phi_n(i)\phi_n(i)^T$$

Then
$$\mu_{min}\{P(t)^{-1}\} \geq \mu_{min}\{\sum_{i=1}^{t} [\prod_{j=i+1}^{t} \lambda(j)]\phi_n(i)\phi_n(i)^T\}$$
$$\geq \mu_{min}\{\sum_{i=t-p+1}^{t} [\prod_{j=i+1}^{t} \lambda(j)]\phi_n(i)\phi_n(i)^T\}, \; t \geq p$$
$$\geq \alpha_2\, \delta^p = p_{min} \geq 0$$

Note that $\mu_{min}\{P(t)^{-1}\} > 0$ since $\mu_{max}\{P(t)\}$ is bounded.

part (ii).

Consider the matrix sequence $\{F(t)^{-1}\}$ defined by

$$F(t)^{-1} = \lambda(t)\, F(t-1)^{-1} + I/p$$

with $F(0)^{-1} = P(0)^{-1}$.

Then $\mathrm{tr}\, F(t)^{-1} = \lambda(t)\mathrm{tr}\, F(t-1)^{-1} + 1$

Also note that $P(t)^{-1} = \lambda(t)\, P(t-1)^{-1} + \phi_n(t)\phi_n(t)^T$

$$\Rightarrow \mathrm{tr}\, P(t)^{-1} \leq \lambda(t)\mathrm{tr}\, P(t-1)^{-1} + \|\phi_n(t)\|^2$$

Since $\|\phi_n(t)\|^2 \leq 1$,

$$\mathrm{tr}\, P(t)^{-1} \leq \lambda(t)\, \mathrm{tr}\, P(t-1)^{-1} + 1$$

$$\Rightarrow \mathrm{tr}\, P(t)^{-1} \leq \mathrm{tr}\, F(t)^{-1}$$

We now want to establish an upperbound on $\mathrm{tr}\, F(t)^{-1}$

Clearly

$$\mathrm{tr}\, F(t)^{-1} = [\prod_{i=1}^{t} \lambda(i)]\mathrm{tr}\, F(0)^{-1} + \sum_{i=1}^{t} [\prod_{j=i+1}^{t} \lambda(j)]$$

It can be shown that

$$\mathrm{tr}\, F(t)^{-1} \leq \frac{1}{\epsilon} + [\prod_{i=1}^{t} \lambda(i)]\mathrm{tr}\, F(0)^{-1} = p_{max} + [\prod_{i=1}^{t} \lambda(i)]\mathrm{tr}\, F(0)^{-1}$$

where $p_{max} = 1/\epsilon$.

But $\mathrm{tr}\, F(t)^{-1} = \lambda(t)\mathrm{tr}\, F(t-1)^{-1} + 1$

$$= [\prod_{i=1}^{t} \lambda(i)]\mathrm{tr}\, F(0)^{-1} + \sum_{i=1}^{t} [\prod_{j=i+1}^{t} \lambda(j)]$$

$$\Rightarrow \sum_{i=1}^{t} [\prod_{j=i+1}^{t} \lambda(j)] \le p_{max}$$

Also since $\mu_{max}\{A\} \le \mathrm{tr}\, A$ for any matrix A,

$\mu_{max}\{P(t)^{-1}\} \le p_{max}$

part (iii).

We know that $\theta(t) = \theta(t-1) - P(t)\phi_n(t)e(t)$

Since $e(t) := y(t) - \phi_n(t)^T\hat{\theta}(t-1) = \phi_n(t)^T\theta(t-1) + \eta^n(t)$

$$\Rightarrow \theta(t) = \theta(t-1) - P(t)\phi_n(t)\phi_n(t)^T\theta(t-1) - P(t)\phi_n(t)\eta^n(t) \qquad (A.7)$$

$$\Rightarrow P(t)^{-1}\theta(t) = \lambda(t)P(t-1)^{-1}\theta(t-1) - \phi_n(t)\eta^n(t) \qquad (A.8)$$

$$\cdots$$

$$= [\prod_{i=1}^{t} \lambda(i)]P(0)^{-1}\theta(0) - \sum_{i=1}^{t} [\prod_{j=i+1}^{t} \lambda(j)]\phi_n\eta^n(i)$$

Since ϕ_n is independent of $\eta^n(i)$ and since $P(t)^{-1}$ is $\mathcal{F}(t-1)$ measurable

$$E\{P(t)^{-1}\theta(t)/\mathcal{F}(t-1)\} = P(t)^{-1}E\{\theta(t)/\mathcal{F}(t-1)\}$$

$$= [\prod_{i=1}^{t} \lambda(i)]P(0)^{-1}\theta(0)$$

Since $\lambda(t)$ is computed at each interval to maintain tr P(t) = constant, P(t) is bounded.

$$\lim_{t\to\infty} E\{\theta(t)/\mathcal{F}(t-1)\} = \lim_{t\to\infty} P(t)[\prod_{i=1}^{t} \lambda(i)]P(0)^{-1}\theta(0) = 0 \text{ a.s.}$$

part (iv).

From (A.7) and (A.8)

$$\theta(t)^T P(t)^{-1}\theta(t) = (\theta(t-1)^T - \theta(t-1)^T\phi_n(t)\phi_n(t)^T P(t)$$
$$-\eta^n(t)\phi_n(t)^T P(t))(\lambda(t)P(t-1)^{-1}\theta(t-1)$$
$$+ \phi_n(t)\eta^n(t))$$
$$= \lambda(t)\theta(t-1)^T P(t-1)^{-1}\theta(t-1)$$
$$-\lambda(t)\phi_n(t)^T P(\cancel{t})P(t-1)^{-1}\theta(t-1)(\phi_n(t)^T\theta(t-1))$$

$$+ \phi_n(t)^T P(t)\phi_n(t)\eta^n(t)^2 + Z(t-1)\eta^n(t)$$

where $Z(t-1)$ contains all the terms $F(t-1)$ measurable multiplying $\eta^n(t)$.

Since $P(t)\phi_n(t) = \dfrac{P(t-1)\phi_n(t)}{\lambda(t) + \phi_n(t)^T P(t-1)\phi_n(t)}$

we have

$$\begin{aligned}\theta(t)^T P(t)^{-1}\theta(t) = {} & \lambda(t)\theta(t-1)^T P(t-1)^{-1}\theta(t-1) \\ & - \frac{\lambda(t)(\phi_n(t)^T\theta(t-1))^2}{\lambda(t) + \phi_n(t)^T P(t-1)\phi_n(t)} \\ & + \frac{\phi_n(t)^T P(t-1)\phi_n(t)}{\lambda(t) + \phi_n(t)^T P(t-1)\phi_n(t)}\eta^n(t)^2 \\ & + Z(t-1)\eta^n(t)\end{aligned}$$

$$\begin{aligned}\Rightarrow \theta^T P(t)^{-1}\theta(t) \le {} & \lambda(t)\theta(t-1)^T P(t-1)^{-1}\theta(t-1) \\ & + \frac{\phi_n(t)^T P(t-1)\phi_n(t)}{\lambda(t) + \phi_n(t)^T P(t-1)\phi_n(t)}\eta^n(t)^2 + Z(t-1)\eta^n(t)\end{aligned}$$

Hence $E\{\theta(t)^T P(t)^{-1}\theta(t)\} \le \lambda(t)E\{\theta(t-1)^T P(t-1)^{-1}\theta(t-1)\} + \gamma^2$

$$\begin{aligned}\Rightarrow E\{\theta(t)^T P(t)^{-1}\theta(t)\} \le {} & \Big[\prod_{i=1}^{t} \lambda(i)\Big]E\{\theta(0)^T P(0)^{-1}\theta(0)\} \\ & + \gamma^2\sum_{i=1}^{t} \Big[\prod_{j=i+1}^{t} \lambda(j)\Big]\end{aligned}$$

From part (ii), $\mu_{min}\{P(t)^{-1}\}\ E\{\|\theta(t)\|^2\} \le \gamma^2\ p_{max}$

From part (i) therefore it follows that

$$E\{\|\theta(t)\|^2\} \le \frac{p_{max}}{p_{min}}\ \gamma^2$$

Appendix 5.2

This appendix contains a collection of simulation examples designed to illustrate the problems associated with the ordinary recursive least squares estimation scheme and the solutions proposed in the ILS algorithm to solve these problems. The simulation examples are presented in sequence, first to demonstrate the problems and second to illustrate the solutions incorporated in ILS. An overview of the simulation examples is presented in Table 5.1. Table 5.2 and Table 5.3 contain a list of process models and controllers used in the simulations.

Table 5.1 An overview of the simulated example runs

Feature	Run #														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Problems:															
Turn-off	x														
Covariance Windup		x	x	x											
Parameter Drifting			x	x					x						
Bursting			x	x											
Bias					x										
ROM Errors								x							
Numerical problems						x	x								
Lack of excitation									x						
Solutions:															
d.c. bias elimination										x					
Scaling											x	x			
Low-pass filtering													x		
Constant trace							x		x			x		x	x
On/off														x	
Comparison:															
Const. tr, λ(t)															x
Const. tr, Resetting															x
Parameters:															
tr P															x

Table 5.2 A list of process models used in the simulations

No.	Process model A(q)Y = B(q)U + C(q)ξ + d	Used in run no.
1	A=q-0.9, B=0.1, C=1, d=0	1
2	A=q-0.9, B=0.1, C=0, d=0	2
3	A=q**2-1.5q+0.7, B=q+0.5, C=q**2-1.5q+0.7	3, 9, 14
4	A=q-0.9, B=0.1, C=q+0.8, d=0	4
5	A=q-0.9, B=0.1, C=0, d=1.0	5, 10
6	A=q-0.9, B=0.01, C=q-0.3, d=0	6, 7, 11, 12
7	A=q**4-2.39q**3+2.691q**2-1.966q+0.6703, B=0.006785q**3+0.06322q**2+0.05831q+0.005334, C=0, d=0	8, 13
8	A=q**2-1.5q+0.7, B=q+0.5, C=q**2-q+0.2, d=0	15

Table 5.3 A list of controllers used in the simulations

Controller	Examples where the controller is used
Dead-beat controller	3, 9, 14
Pole-placement controller	2
Minimum variance controller	4, 6, 7, 11, 12

Example 1

Description

This example illustrates the problem of turn-off.

Definition

Model: 1st order plant model, #1 (Table 5.2)

Identification Method: RLS with $\lambda(t)=1\ \forall t$

Initial Conditions: $\hat{\theta}(0)=(0,0)$; $P(0)=10I$ (tr $P(0)=20$)

Excitation: open-loop; PRBS input of amplitude ±0.5, length 127

Specific points: one parameter of each of the A and B polynomials was estimated; the B-polynomial parameter was changed from 0.1 to 0.5 at $t=401$

Results

Initially P was large. So the parameter estimates converged quickly to values close to true values. However, as time increased, $P \rightarrow 0$ (Figure 5.2) and the estimator slowly turned off. For example, the estimator was not able to track the parameter change at $t=401$ (Figure 5.1). The trace of the covariance matrix is shown in Figure 5.2. If P(0) were smaller, the identification could turn-off before the parameters converged.

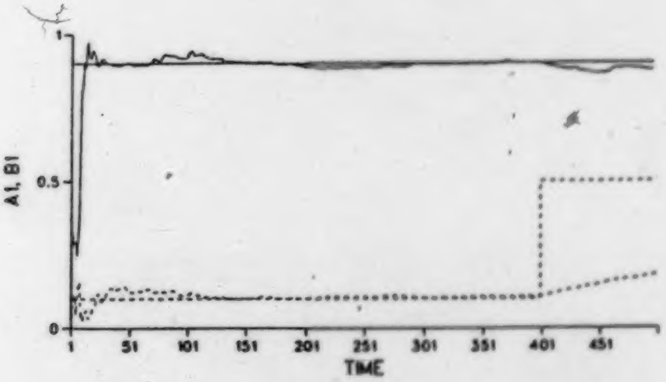


Figure 5.1 Effect of turn-off on the parameter estimates

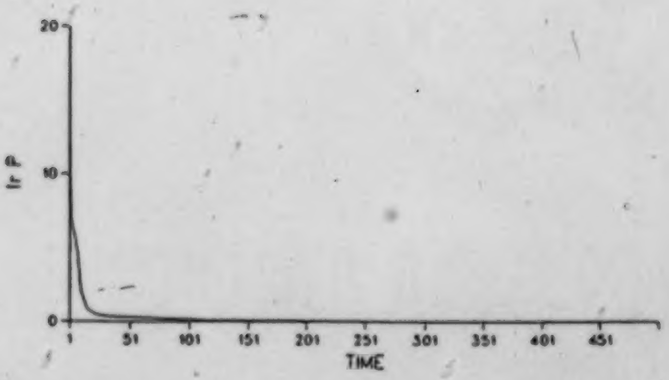


Figure 5.2 Trace of P when turn-off occurs

Example 2

Description

This example illustrates the problem of covariance windup (or blowup).

Definition

Model: 1st order plant, #2 (Table 5.2)

Controller: pole-placement with the closed-loop relation

$y(t) - 0.7y(t-1) = 0.3y_{ref}(t)$

Identification Method: RLS with $\lambda(t)=0.95\ \forall t$

Initial Conditions: $\hat{\theta}(0)=(0,1)$; $P(0)=20I$ ($\mathrm{tr}\ P(0)=40$)

Excitation: closed-loop; changes in y_{ref} (in this case y_{ref} was a square wave of amplitude 1 and period 150Δ, where Δ is the sampling interval)

Results

The system output is shown in Figure 5.3 and indicates good control performance. The covariance matrix grew exponentially (Figure 5.4) when there was no excitation in the system (in the present case this occurred between the reference input changes).

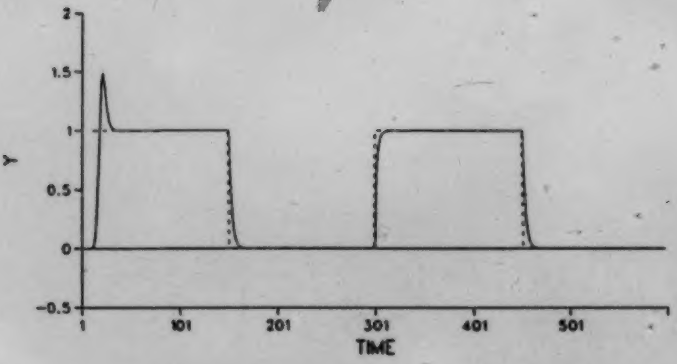


Figure 5.3 System output

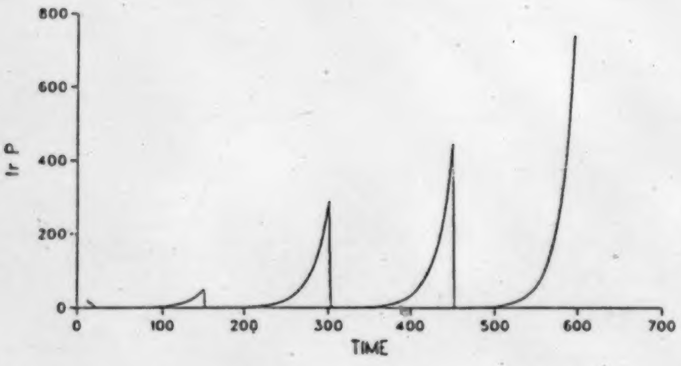


Figure 5.4 Covariance windup with a constant λ

Example 3

Description

This example illustrates the problems of parameter drifting and bursting (in closed-loop) caused by covariance windup.

Definition

Model: 2nd order oscillatory process; #5 (Table 5.2)

Controller: dead-beat

Identification Method: RLS with $\lambda(t)=0.9 \ \forall t$

Initial Conditions: $\hat{\theta}(0)=(0,0,0.1,0)$, $P(0)=I$ (tr $P(0)=4$)

Excitation: closed-loop; step changes in the reference signal which are obvious in Figure 5.5; noise

Specific points: only the parameters of the A and B polynomials (two of each polynomial) were estimated; $var\{\xi(t)\}=0.0004$

Results

1) After each setpoint change the parameter estimates converged rapidly to some reasonable values, but then covariance windup occurred due to poor excitation in the system (Figure 5.7).
2) The noise in the system then caused parameter drifting (Figure 5.6).
3) When the parameter estimates drifted in to regions in which the closed-loop became unstable, bursting occurred (Figure 5.5, at t=700, 880 and 1020).

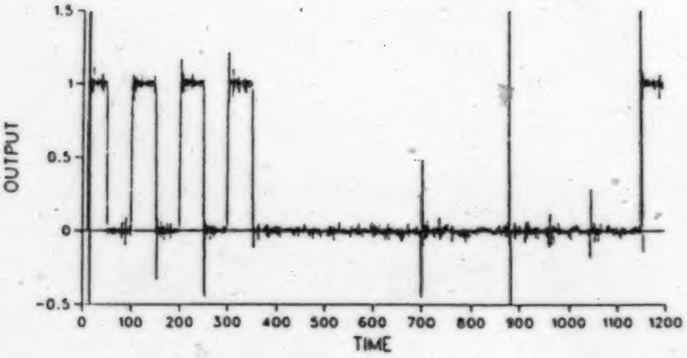


Figure 5.5 System output showing bursts

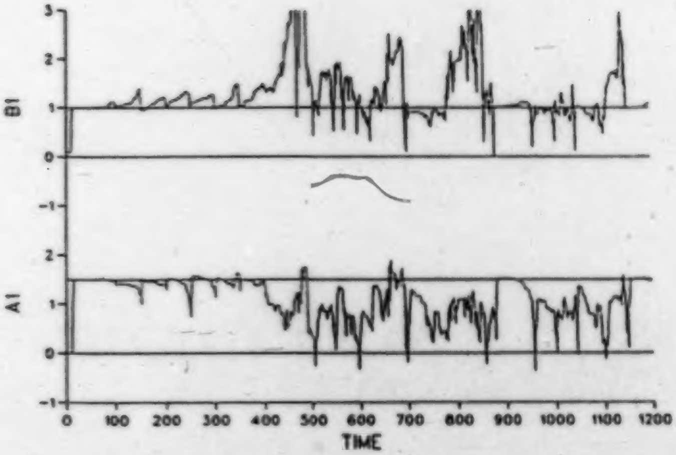


Figure 5.6 Parameter estimates obtained with covariance windup

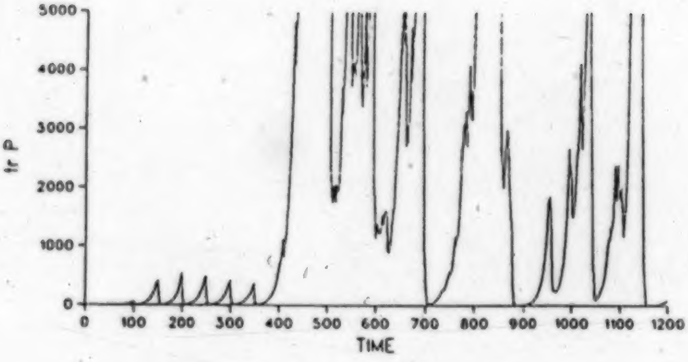


Figure 5.7 Covariance windup with a constant λ

Example 4

Description

This example illustrates that covariance windup, parameter drifting and bursting can occur when the variable forgetting factor due to Fortescue *et al.* (1981) is used in the stochastic case.

Definition

Model: 1st order plant; #3 (Table 5.2)

Controller: minimum variance

Identification Method: RLS with $\lambda(t)$ due to Fortescue *et al.* (1981)

Initial Conditions: $\hat{\theta}(0)=(0,1,1)$, $P(0)=30I$ ($\mathrm{tr}\ P(0)=90$)

Excitation: closed-loop; excitation via noise entering the system

Specific points: one each of the A, B and C polynomial parameters was estimated

Results

1) Noise in the process caused $\lambda(t)<1\ \forall t$ which resulted in covariance windup (Figure 5.10).

2) Consequent parameter drifting (Figure 5.9) caused bursting in y (Figure 5.8 at $t\cong1600$).

3) Tendency to windup was however much reduced compared to a fixed λ.

4) Note that P had no upper or lower bounds. The latter implies that the algorithm could potentially turn-off; tr P decreased sharply due to bursting at $t\cong1600$.

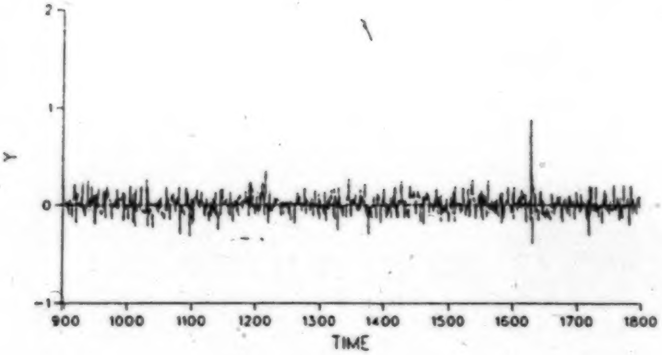


Figure 5.8 System output showing bursts

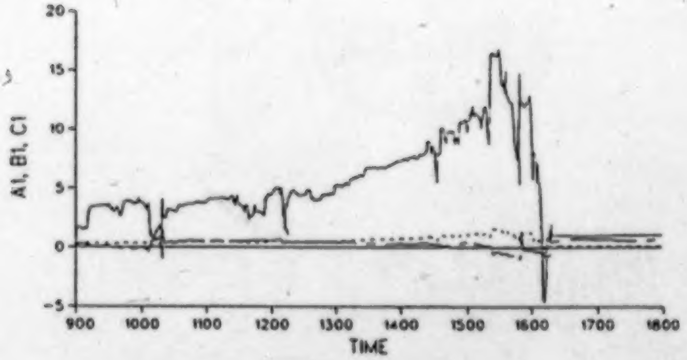


Figure 5.9 Parameter estimates obtained with covariance windup

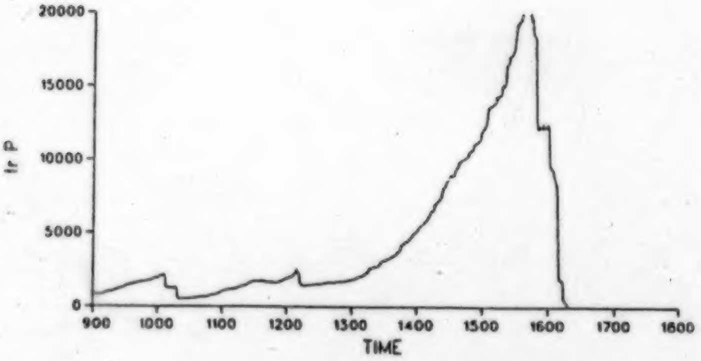


Figure 5.10 Covariance windup with variable forgetting factor of Fortescue

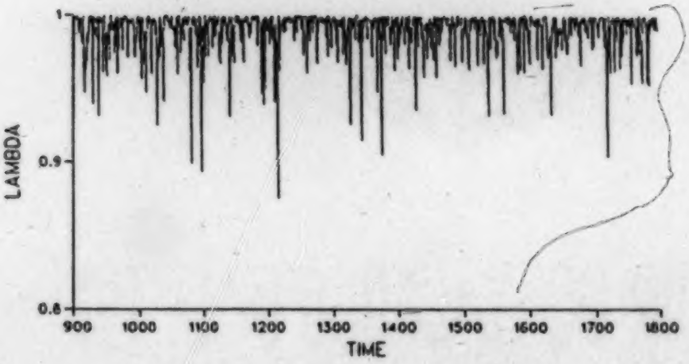


Figure 5.11 Variable forgetting factor of Fortescue

Example 5

Description

This example illustrates the problem of bias in the parameter estimation caused by the presence of an independent, external signal (specifically a d.c. offset or bias).

Definition

Model: 1st order plant; #4 (Table 5.2)

Identification Method: RLS with $\lambda(t)=0.95\ \forall t$

Initial Conditions: $\hat{\theta}(0)=(0,0)$, $P(0)=10I$ (tr $P(0)=20$)

Excitation: open-loop; PRBS input of amplitude ±0.5, length 127

Specific points: one A and one B polynomial parameter were estimated

Results

1) The output was non-zero (i.e. 10) for a zero input due to d.c. bias.
2) The bias in the output signal caused a bias in the estimated parameters (Figure 5.12)
3) Compare this example with example 10 where the bias in the parameters was eliminated through the use of zero-mean deviations in the estimation.

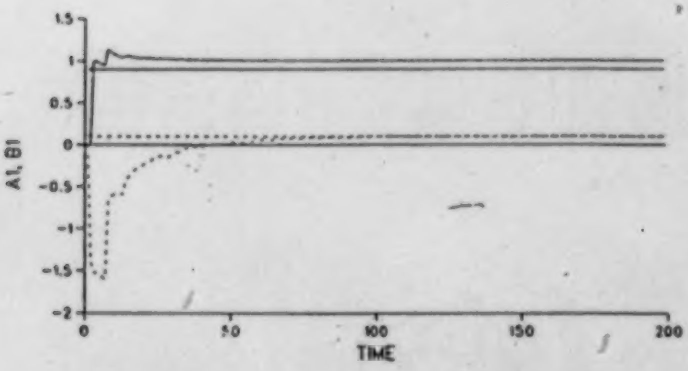


Figure 5.12 Effect of d.c. bias on estimated parameters in RLS

Example 6

Description

This example demonstrates numerical problems caused by poor data in RLS estimation.

Definition

Model: 1st order plant; #6 (Table 5.2)

Controller: minimum variance

Identification Method: RLS with $\lambda(t)=0.95\ \forall t$

Initial Conditions: $\hat{\theta}(0)=(0,\ 1,\ 0)$; $P(0)=10I$ (tr $P(0)=30$)

Excitation: closed-loop; in the present case $y_{ref}=0,\ \forall t$; excitation in the form of noise entering the system

Specific points: one parameter of each of the A, B and C polynomials was estimated; $\mathrm{var}\{\xi(t)\} = 0.25$

Results

1) var(U)>>var(Y) due to the small b_1 parameter and the dead-beat controller used; As a result the P matrix was poorly conditioned.

2) The parameter estimates exhibited increased variations (Figure 5.13), particularly for t>400.

3) Compare this example with example 11, where scaling was used to minimize C{P}

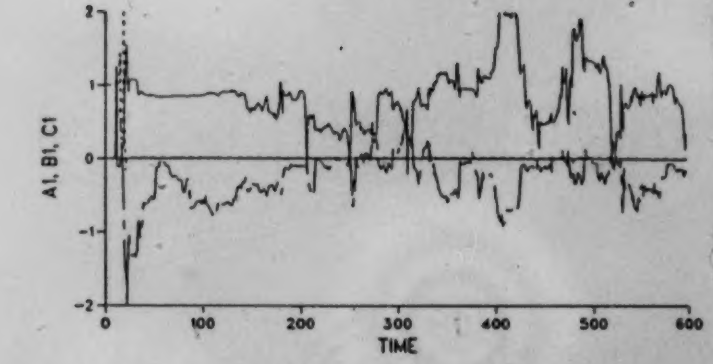


Figure 5.13 Effect of an ill-conditioned P on estimated parameters in RLS

Example 7

Description

This example demonstrates the effect of an ill-conditioned P matrix with ILS algorithm (constant trace without data preprocessing and on/off features which means that it is simply RLS with a constant trace)

Definition

Model: 1st order plant; #6 (Table 5.2)

Controller: minimum variance

Identification Method: ILS (constant trace) with tr P=3

Initial Conditions: $\hat{\theta}(0)=(0,1,0)$

Excitation: closed-loop; noise entering the system

Specific points: one parameter of each of the A, B and C polynomials was estimated

Results

1) The small b_1 parameter resulted in var(U)>>var(Y); As a result the P matrix was badly conditioned and the parameter estimates exhibited a lot of undesired variation (Figure 5.14).
2) Compare this example with example 12 where the same run was repeated with optimal scaling which gives minimum $C\{P_{s(t)}\}$.

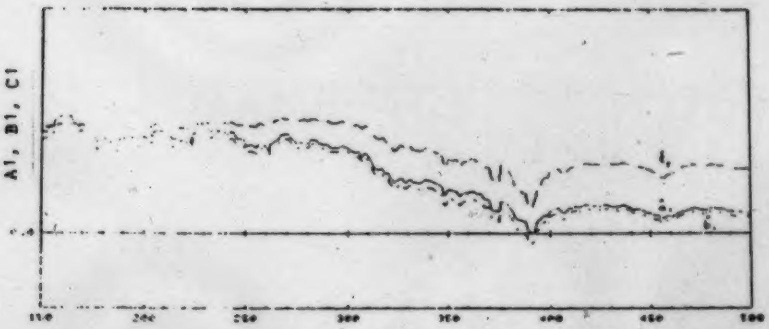


Figure 5.14 Effect of an ill-conditioned P on the estimated parameters in ILS without scaling

Example 8

Description

This example illustrates what happens when a reduced order model is fit to the data from a higher order plant.

Definition

Model: 4th order plant; #7 (Table 5.2)

ID Method: RLS with $\lambda(t)=0.98\ \forall t$

Initial Conditions: $\hat{\theta}(0)=(0,0,0,0)$; $P(0)=10I$ (tr $P(0)=40$)

Excitation: open-loop; PRBS input of amplitude ±0.5 and length 127

Specific points: 2 parameters of each of the A and B polynomial were estimated; The plant has poles at 0.81, 0.9987, 0.2903+i0.8594 and 0.2903-i0.8594; The last two are resonant (or fast) modes

Results

1) The fast modes when excited by the input caused drifting of the parameter estimates (Figure 5.15); notice the significant variations in the parameter estimates even though there is no noise in the plant.
2) The poles of the estimated reduced order model did not correspond to any of the actual poles.
3) Compare this example with example 13 where filtering of the data was employed to cut-off the high-frequency components of the signals.

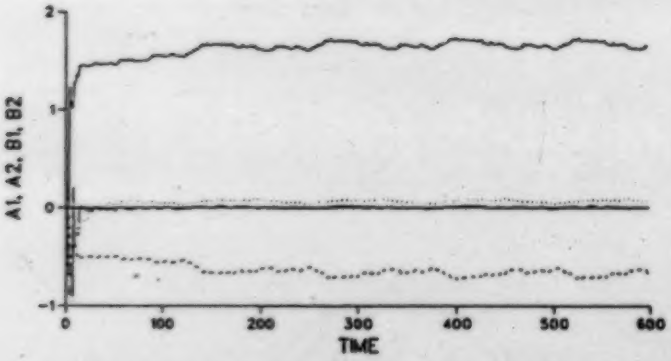


Figure 5.15 Effect of model process mismatch on the estimated parameters in RLS

Example 9

Description

This example demonstrates the problem caused by lack of persistency of excitation.

Definition

Model: 2nd order plant; #3 (Table 5.2)

Controller: dead-beat

Identification Method: ILS (constant trace) without on/off; tr P=4 ∀t

Initial Conditions: $\hat{\theta}(0)=(0,0,0.1,0)$; $P(0)=I$ (tr $P(0)=4$)

Excitation: closed-loop; changes in y_{ref} (cf. Figure 5.16); noise entering the system

Specific points: 2 parameters of each of the A and B polynomials was estimated; $\mathrm{var}\{\xi(t)\}=0.0004$

Results

1) Constant trace for P solved the problem of turn-off and covariance windup.

2) However, when the input/output data were not persistently exciting the parameter estimates could exhibit drifting (Figure 5.17); this phenomena is evident as can be seen from the figure in the interval (400,1150).

3) Continued estimation in such conditions over very long periods could cause closed-loop bursting; Note that this problem although similar to that caused by covariance windup, has an entirely

different origin; the system output is shown in Figure 5.16.

4) Compare this example with example 14 which includes on/off.

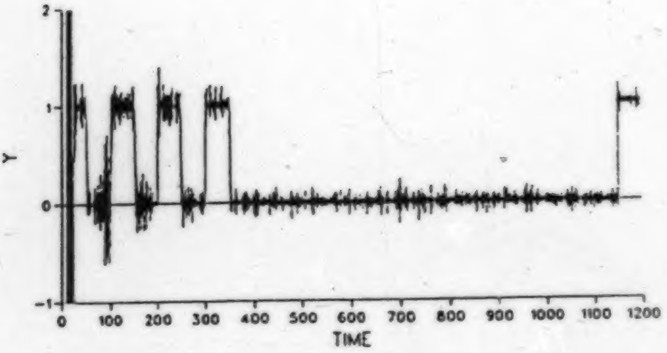


Figure 5.16 System output

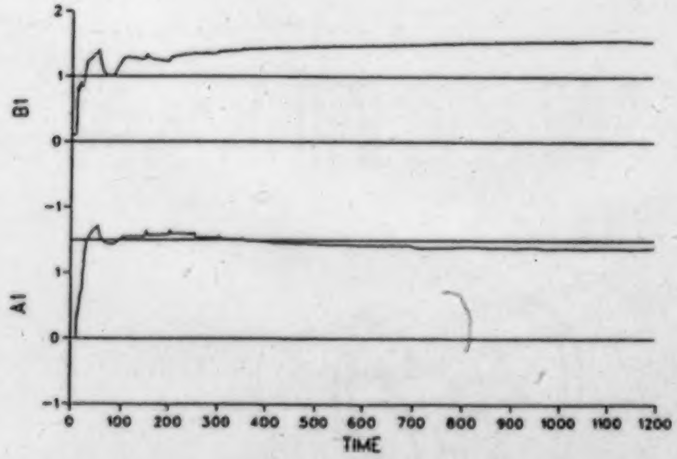


Figure 5.17 Parameter drifting due to lack of excitation

Example 10

Description

This example illustrates the effect of d.c. value elimination.

Definition

Model: 1st order plant; # 5 (Table 5.2)

Identification Method: RLS with $\lambda(t)=0.95\ \forall t$ using the zero-mean deviations obtained by subtracting the d.c. values from the signals

Initial Conditions: $\hat{\theta}(0)=(0,0)$; $P(0)=10I$ ($\mathrm{tr}\ P(0)=20$)

Excitation: open-loop; PRBS input of amplitude ± 0.5 and length 127

Specific points: one A and one B polynomial parameter was estimated

Results

1) The bias in the estimated parameters disappeared (Figure 5.18).

2) Compare this example with example 5.

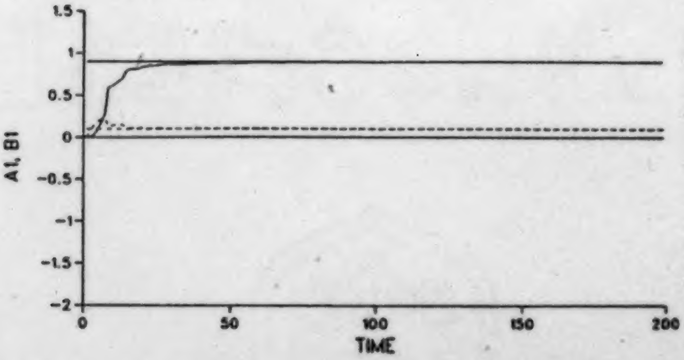


Figure 5.18 Estimated parameters when mean-deviational data are used in RLS

Example 11

Description

This example illustrates the effect of data scaling on RLS estimation.

Definition

Model: 1st order plant; #6 (Table 5.2)

Controller: minimum variance

Identification Method: RLS with $\lambda(t)=0.90\ \forall t$ and optimal scaling

Initial Conditions: $\hat{\theta}(0)=(0,1,0)$; $P(0)=10I$ ($\mathrm{tr}\ P(0)=30$)

Excitation: closed-loop; noise entering the system

Specific points: one parameter of each of the A, B and C polynomial was estimated

Results

1) The scaling improved the parameter estimates which are shown in Figure 5.19.
2) Compare this example with example 6 for $t > 400$ which shows that scaling resulted in smoother estimates (i.e. less variation).

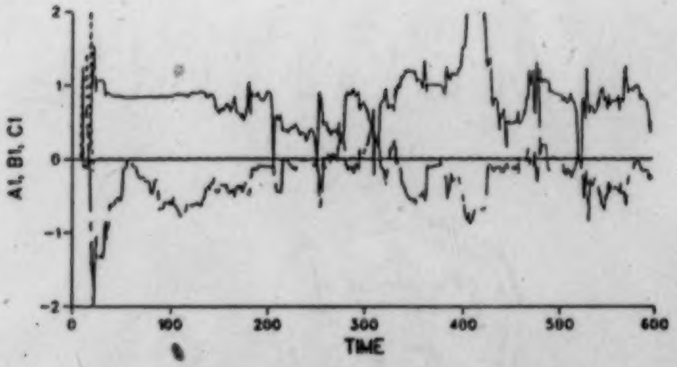


Figure 5.19 Parameter estimates obtained using RLS with data scaling

Example 12

Description

This example illustrates the effect of scaling in ILS (with constant tr and without any other data preprocessing or on/off)

Definition

Model: 1st order plant; #6 (Table 5.2)

Controller: minimum variance

Identification Method: ILS (with constant tr and data scaling)

Initial Conditions: $\hat{\theta}(0)=(0,1,0)$

Excitation: closed-loop; noise entering the system

Specific points: one parameter of each of the A, B and C polynomials was estimated; tr P = 3

Results

1) The scaling matrix was recomputed whenever $C\{P_{s(t)}\} > 100$.
2) After the initial transient, the parameter estimates converged to near constant values (Figure 5.20).
3) Compare this example with example 7 which shows scaling in ILS results in significantly improved (i.e. smoother) parameter estimates.

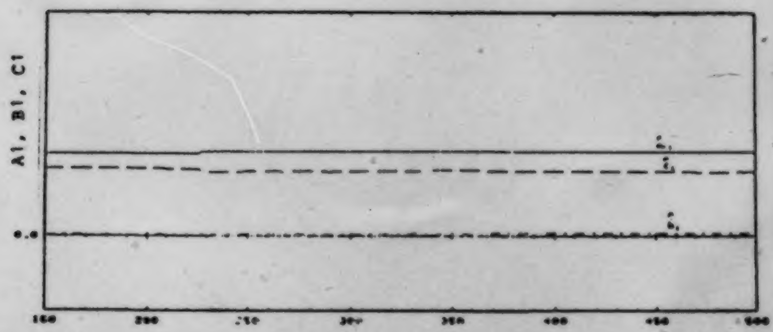


Figure 5.20 Parameter estimates obtained using ILS without the on/off criteria to show the effect of scaling

Example 13

Description

This example illustrates the effect of low-pass filtering of the data.

Definition

Model: 4th order plant; #7 (Table 5.2)

Identification Method: RLS with low-pass filtering of the data and $\lambda(t)=0.98\ \forall t$

Initial Conditions: $\hat{\theta}(0)=(0,0,0,0)$; $P(0)=10I$ (tr $P(0)=40$)

Excitation: open-loop; PRBS input of amplitude ±0.5 and length 127

Specific points: the low pass filter had the transfer function $0.4q/(q+0.6)$

Results

1) The drifting in the estimated parameters was significantly reduced (see Figure 5.21).
2) The poles of the estimated reduced order model corresponded very closely with the dominant poles of the process.
3) Compare versus Example 8.

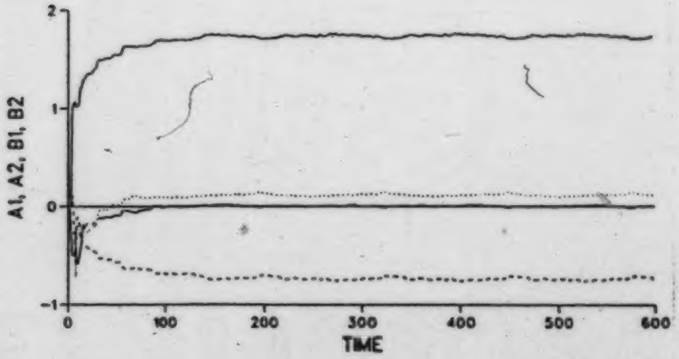


Figure 5.21 Effect of low-pass filtering on the estimated parameters in RLS

Example 14

Description

This example illustrates the effect of the on/off criteria.

Definition

Model: 2nd order plant; #3 (Table 5.2)

Controller: dead-beat

Identification Method: ILS

Initial Conditions: $\hat{\theta}(0)=(0,0,0.1,0)$; $P(0)=I$ (tr $P(0)=4$)

Excitation: closed-loop; changes in y_{ref}; noise in the system

Specific points: two parameters of each of the A and B polynomials were estimated; $d=2$; tr P=constant=4; the scaling matrix was changed whenever $C\{P_{S(t)}\} > 15$; estimation was stopped whenever $C\{P_S\} > 15$; ι was set to 0; $\text{var}\{\xi(t)\}=0.0004$

Results

1) The parameter drifting was almost completely eliminated (Figure 5.23); the system output is shown in Figure 5.22.
2) Compare this example with examples 3, 4 and 9
3) The dots in the bottom portion of Figure 5.23 indicate when estimation was on.

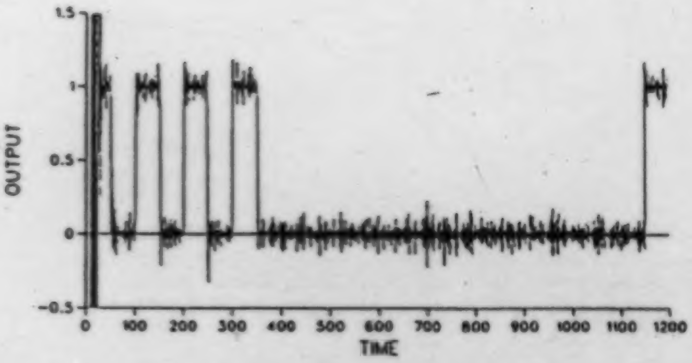


Figure 5.22 System output

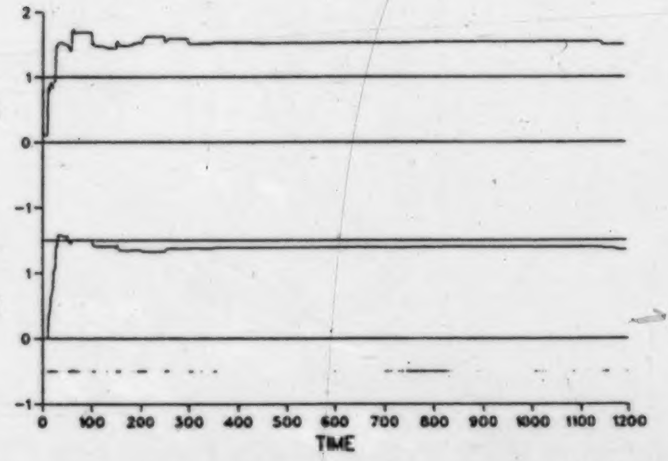


Figure 5.23 Effect of using the on/off criteria in ILS on the estimated parameters

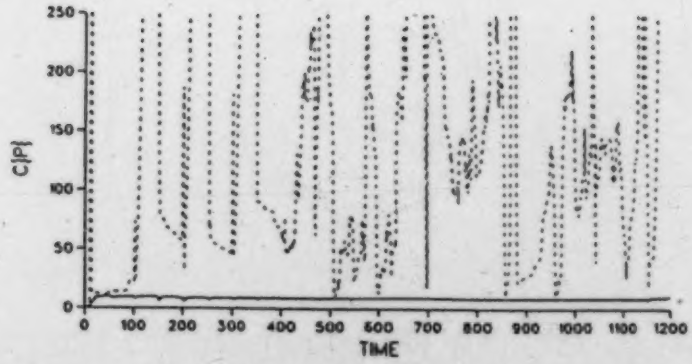


Figure 5.24 Trace of P before (dashed line) and after (continuous line) scaling in ILS with on/off

Example 15

Description

In this example the performance of the variable forgetting factor approach to constant trace proposed in ILS algorithm is compared with the resetting approach to constant trace proposed by Goodwin; Also the the effect of tr P and scaling are demonstrated.

Definition

Model: 2nd order plant; #8 (Table 5.2)

Identification Method: constant trace (ILS and Goodwin)

Initial Conditions: $\hat{\theta}(0)=(0,0,0,0)$; $P(0)=0.25I$ (or I), i.e. tr P=1 (or 4)

Excitation: open-loop; PRBS input of amplitude ± 0.5 and length 127

Specific points: 4 parameters (2 for each of the A and B polynomials) were estimated; standard deviation of $\{\xi(t)\}$ was 0.25

Results

1) Smaller tr P has a filtering or smoothing effect on the parameter estimates (cf. Figure 5.25 versus Figure 5.27).
2) Effect of scaling is not significant when tr P is small but becomes very significant when tr P increases (cf. Figures 5.25 and 5.27 versus Figures 5.26 and 5.28).
3) Larger tr P implies faster convergence (cf. Figure

5.25 versus 5.27 or 5.28).

4) Goodwin's const. tr. algorithm is more noise sensitive than ILS (cf. Figure 5.25 versus 5.29)

5) Goodwin's algorithm was found to diverge in some simulation studies when a larger value for tr P was used; in the present case Goodwin's algorithm gave diverging parameter estimates for even a relatively small trace of 4 where as the ILS approach resulted in good parameter estimates as shown in Figures 5.27 and 5.28; the ILS approach was tested with larger values of tr P and all the runs yielded converging estimates except of course with increased variance due to larger values of tr P.

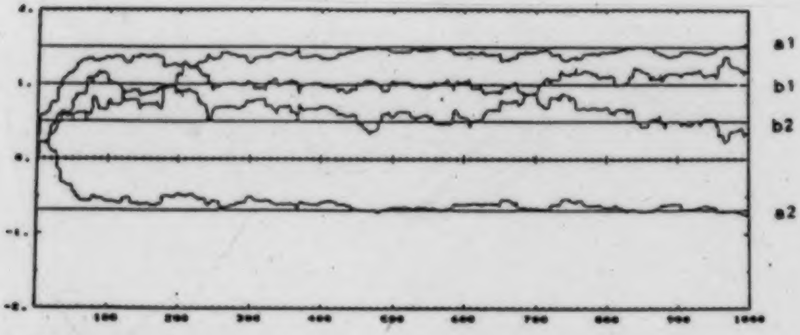


Figure 5.25 Parameter estimates obtained with tr P=1 and without scaling

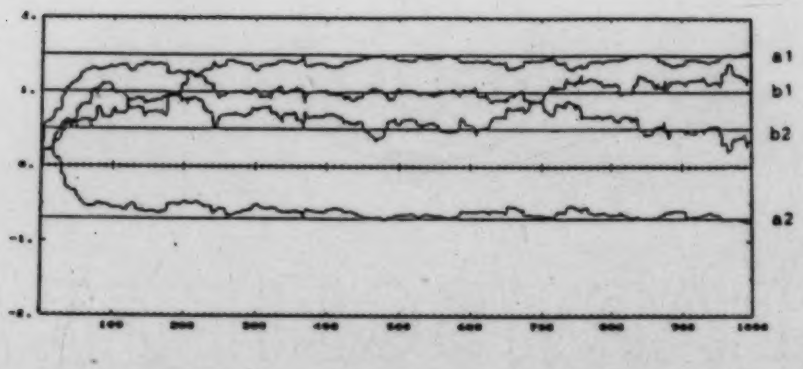


Figure 5.26 Parameter estimates obtained with tr P=1 and with scaling

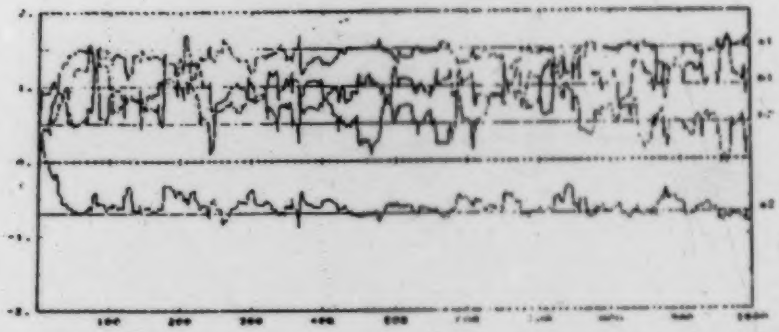


Figure 5.27 Parameter estimates obtained with tr P=4 and without scaling

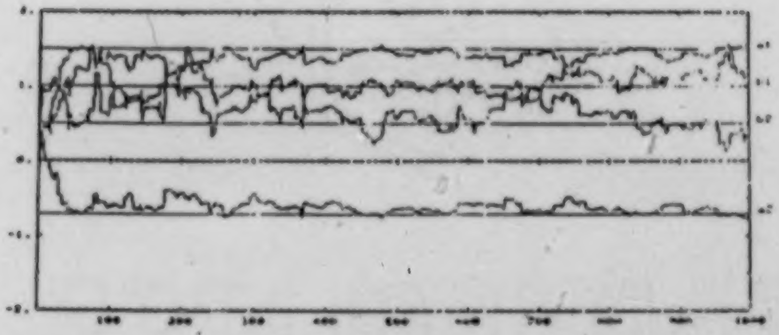


Figure 5.28 Parameter estimates obtained with tr P=4 and with scaling

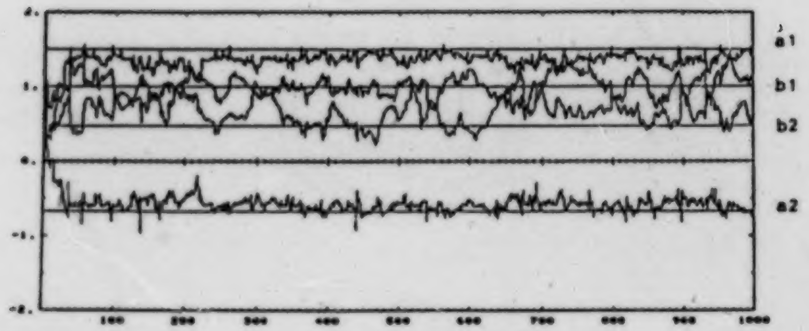


Figure 5.29 Parameter estimates obtained using Goodwins"s constant trace algorithm with tr P=1

6. Multi-step Adaptive Predictive Controller (MAPC) with disturbance modeling and Kalman filter prediction

6.1 Introduction

Adaptive or self-tuning control is a solution proposed to handle the following problems in process-control:

1) poorly known process models;

2) process time-variations;

3) process non-linearities.

A self-tuning controller is simply a combination of a control-law derived by assuming that a model of the process is known, and a parameter estimator to estimate the parameters of the process model (explicit self-tuner) or the parameters of the control-law directly (implicit self-tuner). The idea is that if the parameters converge to the correct values (either of the process or of the control-law) then the performance of the control system approaches that when the true (process) paramters are known. This is referred to as the certainty-equivalence principle. Such a control system though apparently simple presents several problems, the chief one being to ensure overall stability. The various problems and several different approaches to adaptive control have been summarized in a number of excellent review articles (e.g. see Seborg *et al.* (1986), Astrom (1983a), Goodwin *et al.* (1984), Goodwin (1986)).

The major issues to be contended with in the application of adaptive control to industrial processes are handling of:

1) process time-delays;

2) noise ;

3) load-disturbances;

4) parameter estimation;

5) achieving user specified control performance.

These and other important issues in the design of an adaptive controller are briefly discussed below before presenting an overview of the MAPC proposed in this chapter.

Process time-delays

Time-delays are common in process systems and are associated with flows and vary significantly. Therefore they are difficult to specify in advance and cause control performance to deteriorate. One approach to control processes with time-delays is to use predictive control. The Smith predictor (Smith (1957)) is one such algorithm for processes with known delays. Explicit time-domain prediction (forecasting) of the process signals makes it possible to construct a physically realizable controller which compensates for the existence of the process time-delays. In the Smith predictor this is done using a delay-free process model. Self-tuning control algorithms based on prediction such as the self-tuning controller of Clarke and Gawthrop (1979) also follow this approach and can be interpreted as optimal Smith predictors (Gawthrop (1977)). However, single-step predictive control methods are very sensitive to errors in the *a priori* estimate of the time-delay and hence control detuning is often required. An alternative is to use predictive control based on the minimization of a multi-stage cost function.

Process disturbance modeling

Offset-free rejection of load-disturbances with good dynamic behaviour requires the use of a proper load-disturbance internal model (cf. internal model principle). Often the load-disturbances are assumed to change as random steps at random times and although this gives good results, it is not enough to handle more complicated disturbances such as random exponential rises or load-disturbances that affect the process output via a load-disturbance transfer function. The minimum variance strategy handles noise optimally if the correct noise structure is employed in prediction. However, the use of a fixed noise model is often unrealistic as the spectral characteristics of the noise vary significantly with time since the noise results from several disturbance sources. On-line identification of a noise model is usually done in self-tuning control but is associated with convergence difficulties and often gives poor results. A Kalman filter can also be used to generate minimum variance estimates of the process output and has the following advantages:

1) the state-space model used by the Kalman filter allows separate modeling and treatment of process dynamics, noise and load-disturbances;
2) the Kalman filter is easy to tune on-line by changing the noise covariances.

How the disturbances are modelled can influence both model parameter estimation as well as subsequent controller synthesis. The philosophy adopted here is that servo and regulatory control are two separate problems and must be treated as such. Combining both these objectives in control design leads to unnecessary compromises. Good setpoint tracking can be achieved via the open-loop (or feedforward) path if an exact model of the process dynamics is available, e.g. by using an inverse of the process model as a controller. Disturbances and unmodelled effects can then be compensated in a feedback path, using the same inverse model as a controller or a different controller. The two paths complement each other to give asymptotically offset-free tracking and disturbance rejection. The inverse model controller can be implemented as a predictive controller (e.g. see MOCCA, discussed in Chapter 3). The scheme proposed here essentially uses this principle by separately predicting the output of a process model and the combinaton of all other unmodelled effects using a residual model and calculating the control action using a predictive control strategy such that the predicted future outputs are as close to a prescribed reference trajectory as possible in some user defined sense. The advantage of this approach is that it allows easier modeling and estimation of disturbances, compared with existing schemes.

Model identification and parameter estimation

Model identification is an important step in adaptive control and is crucial to its success. The identification scheme used depends on the end use of the model and assumed parameterization of the process and its disturbances. Recursive equation error methods are often used in on-line parameter estimation and one such algorithm which is widely used is the recursive least squares (RLS) or its variant, the recursive extended least squares (RELS). A number of adhoc as well as theoretical modifications are added to the standard parameter estimation methods to enhance their performance in the self-tuning context. These include:

1) data preprocessing (normalization, scaling and filtering);
2) disabling estimation during control saturation and periods of lack of excitation;
3) control of the estimator gain;
4) use of test signals;
5) use of multiple models (Lam (1985)).

Shah and Cluett (1987) review a number of modified least squares estimation schemes. It is proposed in this work to use the improved least squares (ILS) algorithm developed in Chapter 5 as it has many desirable features for use in adaptive control.

The identification scheme requires

1) selection of an appropriate model structure;
2) estimation of selected model parameters.

A number of models have been proposed and used in the literature and a review of these is useful and will give an insight into their advantages and disadvantages for self-tuning control.

6.1.1 A review of models used in self-tuning control

Most of the self-tuning literature starts with the assumption that the process is under sampled-data control and subject to disturbances and is described by the locally linearized model:

$$A(q^{-1})y(t) = q^{-k-1}B(q^{-1})u(t) + \chi(t) \qquad (1)$$

where, A and B are polynomials in the backwardshift operator q^{-1}; k is an assumed or known pure dead-time of the process; y(t) and u(t) are the absolute, measured values of the process output and the process input respectively; and $\chi(t)$ models the disturbances and other effects (e.g. measured or unmeasured disturbances, unmodelled dynamics and d.c. bias) on the process output.

1) Positional or CARMA model

In the self-tuning work of Astrom and Wittenmark (1973) the principal assumption was that the process could be represented as in equation (1) and that the disturbance $\chi(t)$ was stationary with a rational spectral density,

$$\text{i.e. } \chi(t) = C(q^{-1})e(t) \tag{2}$$

where C is a polynomial in q^{-1} and $\{e(t)\}$ is bounded, zero-mean, white noise sequence. This leads to the CARMA (Controlled, Auto-Regressive, Moving-Average) model:

$$A(q^{-1})y(t) = q^{-k-1}B(q^{-1})u(t) + C(q^{-1})e(t) \tag{3}$$

The above model underlies the development of many other methods such as the generalized minimum variance controller (GMV) of Clarke and Gawthrop (1979), the pole placement method of Wellstead *et al.* (1979) and the linear quadratic gaussian (LQG) controller of Lam (1982).

If $\{y(t)\}$ and $\{u(t)\}$ are of non-zero mean, then a linear model is written down in terms of the deviations of the input/output signals from their mean or d.c. values $\overline{y}$ and $\overline{u}$ (which represent the current operating point of the process),

$$\text{i.e. } A(q^{-1})(y(t)-\overline{y}) = q^{-k-1}B(q^{-1})(u(t)-\overline{u}) + C(q^{-1})e(t) \qquad (4)$$

or equivalently,

$$A(q^{-1})y(t) = q^{-k-1}B(q^{-1})u(t) + C(q^{-1})e(t) + d' \qquad (5)$$

The d.c. bias term, d', is usually non-zero and arises whenever a linear model is fit to a non-linear process around an operating point where $\{y(t)\}$ or $\{u(t)\}$ or both have non-zero mean values. It is often assumed constant, but in practice it changes whenever the operating point changes (e.g. step-like changes in the setpoint or load-disturbance).

Equation (5) is often used in place of equation (3) in estimation and controller synthesis. Note that $x(t) := C(q^{-1})e(t) + d'$ in this case. If the disturbance $\{x(t)\}$ is non-stationary (cf. load-disturbances) then the non-stationarity is accounted for by d', when it is taken to be time-varying. However, modeling load-disturbances using d' is very artificial. Also a time-varying d' is difficult to estimate along with the parameters of A, B and C polynomials as d' can sometimes vary faster than these parameters (e.g. when the load-disturbance changes). Note that a constant d.c. bias does not cause any problems in the estimation of A, B and C in equation (5).

2) Incremental or CARIMA model

Recently Tuffs *et al.* (1984) proposed a model where the disturbance process, $\{\chi(t)\}$, is postulated to have stationary increments, i.e.,

$$\Delta\chi(t) = C(q^{-1})e(t) \tag{6}$$

$$\text{or } \chi(t) = C(q^{-1})e(t)/\Delta \tag{7}$$

where $\Delta := (1-q^{-1})$ is the differencing operator. Therefore combining equations (7) and (1) gives the model

$$A(q^{-1})y(t) = q^{-k-1}B(q^{-1})u(t) + C(q^{-1})e(t)/\Delta \tag{8}$$

or equivalently,

$$A(q^{-1})\Delta y(t) = q^{-k-1}B(q^{-1})\Delta u(t) + C(q^{-1})e(t) \tag{9}$$

For a constant d.c. bias, d', $\Delta d'$ is identically zero and hence disappears from the model. This model of the disturbances is adequate to model

1) random steps at random times;
2) brownian motion (or random-walk) type of disturbance such as a drifting stochastic process.

Several self-tuning controllers have been developed based on this incremental or CARIMA (Controlled, Auto-Regressive, Integrated, Moving-Average) model (equation (9)), e.g. LQG (Clarke *et al.* (1987c, 1987d)), and GPC (Clarke *et al.* (1987a, 1987b)).

The major advantages of this model are:

1) there is no need to estimate a d.c. bias;
2) it gives rise to integral action in the controller naturally (cf. internal model principle for eliminating step changes).

The chief disadvantage occurs in estimation as well as in control: the incremental data amplifies the effect of noise in the measured signals (note that the variance of signal increments is twice that of the original signal).

Also $x(t) := C(q^{-1})e(t)/\Delta$ cannot model more complicated load-disturbances (e.g. instead of random steps at random times, random exponential rises or periodic disturbances). In this case a more general model such as $x(t) := C(q^{-1})e(t)/N(q^{-1})$ is needed where N is a load-disturbance internal model polynomial in q^{-1}. A special case of this with $N(q^{-1}) := \Delta$ is the CARIMA model.

3) Model used by M'Saad *et al.* (1986)

M'Saad *et al.* (1986) assume in the development of their adaptive LQG controller that the disturbance process, $\{x(t)\}$, is given by:

$$N(q^{-1})x(t) = F(q^{-1})e(t)/G(q^{-1}) \qquad (10)$$

N, F and G are user chosen polynomials in q^{-1} which are included to represent in equation (1) unmodelled response such as noise, load-disturbances and unmodelled effects. This model is more useful in industrial process environment and can be considered as a generalization of the CARIMA model. Note $N(q^{-1})=0$ when $q^{-1}=1$.

4) Model used by Goodwin (1986)

Goodwin (1986) proposes a model of the following form in the development of an adaptive robust controller:

$$A(q^{-1})N(q^{-1})y(t) = q^{-k-1}B(q^{-1})N(q^{-1})u(t) + C(q^{-1})N(q^{-1})e(t) \qquad (11)$$

where $N(q^{-1})$ is an annihilating or nulling polynomial for determinstic or load-disturbances with zeros on the boundary of the unit circle. If $e(t)=0$, the disturbances are purely determinstic and if $N(q^{-1})=1$, then the disturbances are purely stochastic. Furthermore Goodwin suggests the use of a fixed N polynomial and a low-pass filter to filter out the noise instead of estimating N and C although these can be estimated along with the parameters of A and B.

In summary the above review suggests that appropriate modeling of disturbances is important for estimation; the polynomials associated with the disturbance model are rarely estimated in practice; instead fixed, user chosen approximations are used.

The identification used in the proposed adaptive control scheme is based on the idea of multiple models (a process model and a residual disturbance model). First the parameters of the process model are estimated using a simplified input/output model (e.g. equation (5)), with adequate precautions to give good, unbiased estimates of the parameters of A and B polynomials. Then a residual sequence $\{\hat{x}(t)\}$ is calculated using equation (1) that contains the unmodelled process response. A residual model is identified by fitting a single series model to $\{\hat{x}(t)\}$. These two models can then be directly substituted into a state-space model which is used in the controller synthesis. This two-stage identification avoids the need for assuming a fixed noise observer polynomial or a load-disturbance internal model polynomial and provides a way of including these in a separate residual model; It also avoids the convergence difficulties associated with a single-stage identification where all the model polynomials (A, B, C, N etc.) are estimated in a single step.

Process constraints

Constraints are important in process control and when present they may cause problems in the identification part of an adaptive controller. The proposed control algorithm can easily be extended to include constraints, although in the multivariable case these may require on-line iterative calculations as the control problem then becomes one of constrained optimization. Constraints are handled in the proposed scheme via the supervisory system, which adjusts the reference input to satisfy the constraints. The supervisory system can also be used to generate other types of reference trajectories (e.g. a filtered setpoint).

Using the supervisory system to implement the constraints has the advantage of dividing the control problem into two consecutive tasks: first the calculations on the reference trajectory; and second the calculation of the control action so that the output tracks the computed reference trajectory . One advantage of this separation is that the supervisory control calculations can be done at a different (longer) sampling interval and can be stopped during periods of steady state operation. However, unless the feedback signal (i.e. residual forecast) is also used in the calculations the unmeasured disturbances and or unmodelled dynamics could result in some of the constraints being violated.

AI executive

The two-stage identification method proposed in this work relies on a set of heuristic rules for its success. The expert systems approach is a useful tool to implement such rules. An AI based executive is therefore suggested to supervise parameter estimation and to direct the overall operation of the adaptive controller. The AI executive is useful as an easy-to-use operator interface during commissioning and to ensure a satisfactory performance and stability in abnormal situations of the adaptive controller.

6.1.2 An overview of proposed MAPC

The control algorithm proposed here is based on a state-space model of the process and disturbances (the disturbance model states correspond to an assumed load-disturbance internal model or an on-line estimated residual model). A Kalman filter predictor is used to generate predictions of the process output a number of time steps into the future. The Kalman filter predictor generates asymptotically offset-free predictions of the output if the disturbance model is appropriately chosen. Then a set of control actions over a control horizon is calculated such that the predicted output matches a pre-specified reference trajectory, and minimizes a user specified cost function. The control horizon is less than or equal to the output horizon with the control action increments beyond the control horizon assumed to be zero. This gives a lot of flexibility to the control algorithm. The first control action in the control horizon is implemented and all the control calculations are repeated at the next sampling interval in a receding-horizon or moving-window strategy. The algorithm can be shown to reduce to a number of useful special cases such as minimum variance, generalized minimum variance, Smith predictor, extended horizon adaptive control etc. The advantages of the proposed control scheme are improved robustness in handling time-delays, ability to follow operator specified setpoint or reference trajectories and improved disturbance rejection. A self-tuning version of the control algorithm is obtained by separately identifying an on-line model of the process dynamics plus a residual disturbance model.

In addition to the problems of time-delays and disturbances, a practical adaptive control scheme must be able to:

1) overcome the problems caused by NMP or unstable zeros, which arise in discrete time control due to fast sampling or which may be present inherently in the process;
2) handle unstable and poorly damped processes; although most process systems are characterized by stable, well damped, dead-time dominated dynamics, unstable and/ot complex poles are not uncommon (e.g. reactor control) and can present significant control difficulties.

The proposed control algorithm performs satisfactorily in the above two cases as well as when the process model is over- or under-parameterized (cf. predictive control). Explicit adaptive control schemes such as pole placement and linear quadratic gaussian based controllers are sensitive to over-parameterization of the process model since the estimated model may then contain pole-zero cancellations. However, predictive control methods do not suffer from this problem. Under-parameterization of the process model can result in significant unmodelled dynamics and the proposed scheme handles these via the residual model.

The proposed control algorithm is presented for the SISO case with a preliminary set of simulation examples. Its extension to the multivariable case is left as future work. The design of MAPC involves the following steps:

1) process representation (or model selection for estimation and controller synthesis);
2) model identification and parameter estimation;
3) prediction of the process output
    a) model based prediction using measured input and output signals;
    b) feedback and prediction of the residuals (cf. disturbances);
4) control calculations;
5) the supervisory system calculations;
6) the executive system;

A list of the features of MAPC is presented in Table 6.1 and and can be used as a summary. A high-level schematic block diagram of MAPC is shown in Figure 6.1 and the structure of the rest of the chapter closely follows this

Table 6.1 MAPC Features

Actual process
- SISO/MIMO (non-square)**
- open-loop stable/unstable
- non-minimum phase
- time-varying gain and/or dynamics
- known time-delays (can handle unkovn or varying delay)
- mild non-linearities
- noise
- load-disturbances

Process representation
- multiple models for identification**
- state-space model (in observable canonical form) for controller synthesis
- formulation uses absolute, measured values of the signals directly as opposed to incremental values
- constraints on process variables (i.e. u, y etc.)

Model identification and parameter estimation
- process model (ARMA)
- residual model (single series)**
- ILS**
    - datapreprocessing
        mean elimination
        low-pass filtering
        normalization
        data scaling
    - variable forgetting factor → constant trace
    - on/off criteria (based on measures of excitation)
    - parameter projection or constraints

Output prediction
- multi-step-ahead Kalman filter prediction**

Controller
- multistep predictive control**
- full residual (disturbance) modeling**
- receding-horizon (or moving-window) strategy
- weighted, quadratic cost index

Supervisory system**
- generates the reference input trajectory for use in the predictive controller
- implements hard constraints by changing the reference trajectory;

AI Executive
- startup/commissioning
- mode switching
    - startup
    - emergency
- improved operation
    - supervision of identification (e.g. detection of load-disturbances)
    - selection of parameters (ILS, KFP, controller)
    - discounting the residual model

Other practical aspects
- signal conditioning and validation, e.g. detection of outliers in measurements
- numerically robust implementation through the use of optimal scaling in ILS, KFP and controller calculations

---

Features marked with ** indicate the most important features.

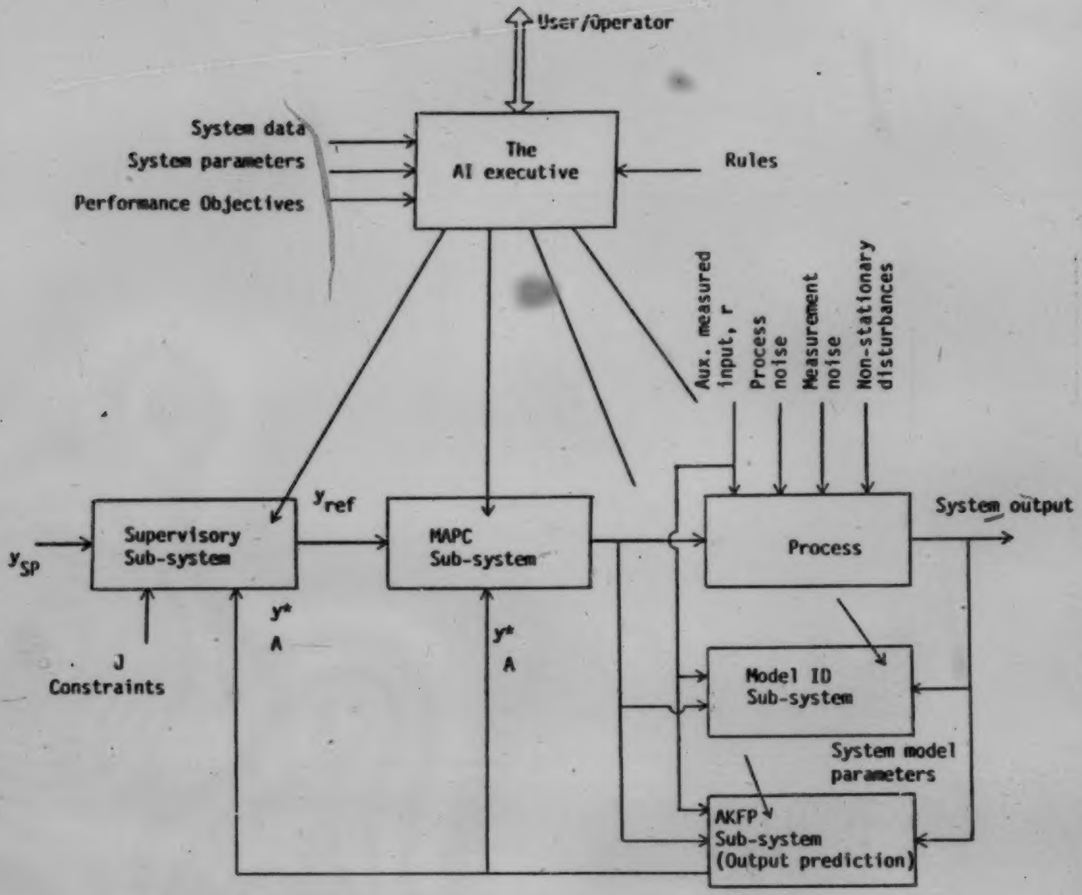

User/Operator
System data
System parameters
Performance Objectives
The AI executive
Rules
Aux. measured input, r
Process noise
Measurement noise
Non-stationary disturbances
y_{SP}
Supervisory Sub-system
y_{ref}
MAPC Sub-system
Process
System output
y^*
A
J
Constraints
y^*
A
Model ID Sub-system
System model parameters
AKFP Sub-system (Output prediction)

Figure 6.1 A high-level, schematic block diagram of MAPC

figure. The next six sections of the chapter discuss the six main steps of the proposed scheme. This is followed by a discussion of the convergence and stabilty issues of MAPC; a brief literature review; and finally some SISO simulated examples to demonstrate its various features.

6.2 Process representation

For the purpose of this presentation it is assumed that the process plus the disturbances can be represented by the following locally linearized model:

$$A'(q^{-1})y(t) = q^{-k-1}B'(q^{-1})u(t) + \chi(t) \tag{12}$$

where as usual y(t) and u(t) are the process output and the process input respectively; k is an assumed or known pure time-delay of the process in number of sampling intervals (the extra delay is due to discrete-sampling and zero-order hold);

$$A'(q^{-1}) := 1 + \sum_{i=1}^{n'_a} a'_i q^{-i} \; ; \; B'(q^{-1}) := \sum_{i=1}^{n'_b} b'_i q^{-i} \tag{13}$$

n'_a and n'_b are assumed or known upper bounds on the orders of the polynomials in the process model (i.e. A' and B'). Sometimes the B' polynomial is over-parameterized, especially when the process dead-time is unkown or is expected to vary significantly so that $n'_b>n'_a$ in some cases. However, in the rest of this chapter it is assumed without loss of generality that $n'_a=n'_b=n'$, where n' represents the order or an upper bound on the order of the process model. $\{x(t)\}$ is a sequence that represents the unmodelled process response, i.e. the effects of load-disturbances, noise and unmodelled dynamics. The following model is assumed for the description of this disturbance sequence:

$$F(q^{-1})x(t) = C'(q^{-1})e(t)/G(q^{-1}) \qquad (14)$$

where $\{e(t)\}$ is a bounded, zero-mean, white noise sequence; F, C' and G are monic polynomials in q^{-1}. C' and G are assumed to be stable (i.e. their zeros are strictly outside the unit circle in the q^{-1} domain). They are introduced for parameter estimation purpose and to model colored, process noise. F is the load-disturbance internal model possibly with zeros on the boundary of the unit circle. It can model anything from a simple d.c. bias to a more general disturbance including periodic disturbances. Note that when $\{x(t)\}$ is a purely determinstic load-disturbance, $F(q^{-1})x(t)=0$ with $F(q^{-1})=0$ at $q^{-1}=1$. $\{x(t)\}$ is a purely stochastic process when $F(q^{-1})=1$. Usually $F(q^{-1})=\Delta D(q^{-1})$, where $\Delta := (1-q^{-1})$ and $D(q^{-1}) := 1 + \sum_{i=1}^{n_d} d_i q^{-i}$ with $D(q^{-1})$ stable. F is required for offset-free prediction and hence offset-free load-disturbance rejection.

The model given by equations (12) and (13) is more suitable for industrial process disturbance description than the usual CARMA or CARIMA models. This model has also been used by M'Saad *et al.* (1986). Note that the CARIMA model is a special case when $F(q^{-1})=\Delta$. The complete process model then becomes

$$A'(q^{-1})y(t) = q^{-k-1}B'(q^{-1})u(t) + C'(q^{-1})e(t)/(F(q^{-1})G(q^{-1})) \quad (15)$$

Cross multiplying by G and defining new polynomials,

$$A(q^{-1}) := A'(q^{-1})G(q^{-1}) = 1 + \sum_{i=1}^{n} a_i q^{-i};$$

$$B(q^{-1}) := B'(q^{-1})G(q^{-1}) = \sum_{i=1}^{n} b_i q^{-i};$$

where $n:=n'+n_g$, the model given by equation (14) becomes

$$A(q^{-1})y(t) = q^{-k-1}B(q^{-1})u(t) + C'(q^{-1})e(t)/F(q^{-1}) \quad (16)$$

For simplicity one can assume $G(q^{-1})=1$. To facilitate writing down an equivalent state-space model for use in the controller synthesis in which the states corresponding to the load-disturbances appear separately, the above model is put into the following form:

$$y(t) = q^{-k-1}\frac{B(q^{-1})}{A(q^{-1})}u(t) + \frac{C'(q^{-1})}{A(q^{-1})F(q^{-1})}e(t) \quad (17)$$

$$\text{or } y(t) = q^{-k-1}\frac{B(q^{-1})}{A(q^{-1})}u(t) + \frac{C(q^{-1})}{A(q^{-1})}e(t) + \frac{E(q^{-1})}{F(q^{-1})}e(t)$$

$$= q^{-k-1}\frac{B(q^{-1})}{A(q^{-1})}u(t) + \frac{C(q^{-1})}{A(q^{-1})}e(t) + d(t) \quad (18)$$

where $C'(q^{-1}):=C(q^{-1})F(q^{-1})+E(q^{-1})A(q^{-1})$ and $d(t):=E(q^{-1})e(t)/F(q^{-1})$. Note that the second term on the right hand side is due to noise and d(t) is due to load-disturbances. For complete separation of the process and disturbance states in the state-space model, the load-disturbance is considered as additive at the output. Also note that the polynomial $C'(q^{-1})$ in equation (16) is defined only for the purpose of representation. It is never required in actual implementation of the proposed adaptive controller. There is no need to actually compute C and E from C'. The polynomials C', F and G are abosorbed in the residual model which models the residuals $\{\hat{x}(t)\}$.

From equation (18) one can write down the following CARMA model for estimation of parameters of A and B:

$$A(q^{-1})y(t) = q^{-k-1}B(q^{-1})u(t) + C(q^{-1})e(t) + d'(t) \qquad (19)$$

where $d'(t) := A(q^{-1})d(t)$ can be interpreted as a d.c. bias and eliminated asymptotically from the above equation using mean deviational data. Note that only the sustained or steady state part of d' is eliminated by using estimates of $\bar{u}$ and $\bar{y}$.

For writing down the state-space model, the polynomials C, E and F must be defined. One can take for simplicity $E(q^{-1})=1$ or alternatively think of it as being included in F after long-division and truncation. This means that the load-disturbance is modelled as a single series (cf. auto-regressive) model. Assume,

$$C(q^{-1}) := 1 + \sum_{i=1}^{n_c} c_i q^{-i} \text{ and } F(q^{-1}) := 1 + \sum_{i=1}^{n_f} f_i q^{-i}$$

More generally n_c may be different from n, but is set equal to n here for simplicity. The state-space model corresponding to equation (18) in observable canonical form is then given by:

$$x(t+1) = \Phi_1 x(t) + \Lambda_1 u(t) + \Gamma_1 e(t) \tag{20}$$

$$\xi(t+1) = \Phi_2 \xi(t) + \Gamma_2 e(t) \tag{21}$$

$$y(t) = H_1 x(t) + H_2 \xi(t) + e(t) \tag{22}$$

$$\text{or } y(t) = y_m(t) + d(t) + e(t) \tag{23}$$

where,

$y_m(t) = H_1 x(t)$;

$d(t) = H_2 \xi(t)$;

$x(t) = (x_1(t), \ldots, x_n(t))^T$ is the process state vector;

$\xi(t) = (\xi_1(t), \ldots, \xi_n(t))^T$ is the disturbance state vector;

$$\Phi_1 := \begin{bmatrix} 0 & 0 & \cdots\cdots\cdots & 0 & 0 \\ 1 & 0 & \cdots\cdots\cdots & 0 & 0 \\ \cdot & \cdot & \cdots\cdots\cdots & \cdot & \cdot \\ \cdot & \cdot & \cdots\cdots\cdots & \cdot & \cdot \\ 0 & 0 & \cdots 1 \cdots & 0 & -a_n \\ 0 & 0 & \cdots 0 \ 1 \ . & 0 & -a_{n-1} \\ \cdot & \cdot & \cdots\cdots\cdots & \cdot & \cdot \\ \cdot & \cdot & \cdots\cdots\cdots & \cdot & \cdot \\ 0 & 0 & \cdots 0 \ 0 \ . & 1 & -a_1 \end{bmatrix}$$

$$\Phi_2 := \begin{bmatrix} 0 & 0 & \cdots\cdots\cdots & 0 & -f_{n_\xi} \\ 1 & 0 & \cdots\cdots\cdots & 0 & -f_{n_\xi - 1} \\ \cdot & \cdot & \cdots\cdots\cdots & \cdot & \cdot \\ \cdot & \cdot & \cdots\cdots\cdots & \cdot & \cdot \\ \cdot & \cdot & \cdots\cdots\cdots & \cdot & \cdot \\ 0 & 0 & \cdots\cdots\cdots & 1 & -f_1 \end{bmatrix}$$

$$\Lambda_1 := \begin{bmatrix} b_n \\ b_{n-1} \\ \cdot \\ \cdot \\ b_1 \\ 0 \\ \cdot \\ 0 \end{bmatrix} ; \; \Lambda_2 := [0]$$

$$\Gamma_1 := \begin{bmatrix} 0 \\ 0 \\ \cdot \\ 0 \\ a_n - c_n \\ \cdot \\ \cdot \\ \cdot \\ a_1 - c_1 \end{bmatrix}$$

$$\Gamma_2 := \begin{bmatrix} 0 \\ \cdot \\ \cdot \\ 0 \\ 1 \end{bmatrix}$$

$$H_1 := [0 \; 0 \; \ldots\ldots\ldots \; 0 \; 1]$$

$$H_2 := [0 \; 0 \; \ldots\ldots\ldots \; 0 \; 1]$$

The dimensions of Φ_1, Φ_2, Λ_1, Λ_2, Γ_1, Γ_2, H_1 and H_2 are respectively $(n+k)x(n+k)$, $n_t x n_t$, $(n+k)x1$, $n_t x1$, $(n+k)x1$, $n_t x1$, $1x(n+k)$ and $1xn_t$. The complete process model in state-space form can be written more compactly as:

$$z(t+1) = \Phi z(t) + \Lambda u(t) + \Gamma e(t) \tag{24}$$

$$y(t) = Hz(t) + e(t) \tag{25}$$

where $z(t) = (x(t)^T, \xi(t)^T)^T$;

$$\Phi = \begin{bmatrix} \Phi_1 & 0 \\ 0 & \Phi_2 \end{bmatrix} \quad \Lambda = \begin{bmatrix} \Lambda_1 \\ \Lambda_2 \end{bmatrix} \quad \Gamma = \begin{bmatrix} \Gamma_1 \\ \Gamma_2 \end{bmatrix}$$

and $H = (H_1, H_2)$.

Remark

If $C(q^{-1})=A(q^{-1})$ then the noise is simply additive at the process output (cf. measurement noise in y). The structured or process noise which can be predicted to obtain minimum variance control can be thought of as included in the disturbance model (i.e. the signal $d(t)$). This interpretation is necessary since the residuals in practice cannot be separated into noise and load-disturbances.

Feedforward input

If there is an auxiliary measured disturbance input, $r(t)$, it can be included in the model (i.e. equation (15)) as

$$A(q^{-1})y(t)=q^{-k-1}B(q^{-1})u(t)+L(q^{-1})r(t)+C'(q^{-1})e(t)/F(q^{-1}) \tag{26}$$

Assuming $L(q^{-1}):=1+\Sigma_{i=1}^{n} l_i q^{-i}$, the state-space model then becomes:

$$z(t+1) = \Phi z(t) + \Lambda u(t) + \Lambda' r(t) + \Gamma e(t) \tag{27}$$

$$y(t) = Hz(t) + e(t) \tag{28}$$

where

$$\Lambda' := \begin{bmatrix} 0 \\ \cdot\cdot \\ 0 \\ d_n - a_n \\ \cdot \\ \cdot \\ \cdot \\ d_1 - a_1 \end{bmatrix}$$

with the dimension of Λ' as (n+k)x1. Then equations (26), (27) and (28) can be used as a basis for parameter estimation and for output prediction. Note that in this case the L polynomial can also be estimated. For simplicity in the rest of the chapter such a feedforward input is not used.

Remarks

1) If the parameters of A, B, C and F are known then one can estimate the process and disturbance state vectors (i.e. $x(t)$ and $\xi(t)$ respectively) using a Kalman filter and their contribution to the process output can be obtained by summation. A single Kalman filter or two separate Kalman filters can be used. Note that the state coefficient matrix, Φ, is block-diagonal and hence the process dynamics and disturbances can be treated separately. Using two Kalman filters is numerically more robust and also reduces the computer storage requirements. Such two-stage estimation in Kalman filtering is not uncommon where the bias in the process is separately estimated (see Freidland (1978)).

The bias estimation is equivalent to load-disturbance estimation here.

2) Since C is not usually available *a priori* and in practice it can be included in the residual model, one could use the following stochastic state-space model,

$$z(t+1) = \Phi z(t) + \Lambda u(t) + w(t) \tag{29}$$

$$y(t) = Hz(t) + v(t) \tag{30}$$

where $\{w(t)\}$ and $\{v(t)\}$ are bounded, zero-mean, white noise sequences with $\mathrm{cov}\{w(t)\}:=E\{w(t)w(t)^T\}=R_w$ and $\mathrm{cov}\{v(t)\}=E\{v(t)^2\}=R_v$. $\{w(t)\}$ and $\{v(t)\}$ can then be used to represent noise in the measured signals. Also when the Kalman filter equations are written down for the state-space model given by equations (29) and (30) the covariance of the state estimation error vector and the Kalman gain are explicitly available in terms of R_w and R_v and need not be deduced from the noise polynomial C. Also note that by a suitable choice of the noise covariances, measurement noise can be rejected in the Kalman filter. The structured process noise is however included in the residual disturbance model for prediction.

3) From the stochastic state-space model given by equations (29) and (30) one could deduce the model given by equation (15) via the Kalman filter.

6.2.1 One-step-ahead Kalman filter predictor

A Kalman filter can be used to estimate the states of the stochastic state-space model given by equations (29) and (30). The one-step-ahead state predictor form of the Kalman filter (which is referred to in this work as the one-step-ahead Kalman filter predictor) is given by (Godfrey and Jones (1986); Kumar and Varaiya (1986)):

$$\hat{z}(t+1/t) = \Phi\hat{z}(t/t-1) + \Lambda u(t) + K(t)e(t) \tag{31}$$

$$y(t) = H\hat{z}(t/t-1) + e(t) \tag{32}$$

which is optimal in the sense of minimum variance estimation when the Kalman gain, K(t), is found from:

$$K(t) = \Phi P(t/t-1)H^T(HP(t/t-1)H^T+R_v)^{-1} \tag{33}$$

where the covariance matrix, P(t/t-1), is updated according to:

$$P(t+1/t) = \Phi P(t/t-1)\Phi^T - K(t)HP(t/t-1)\Phi^T + R_w \tag{34}$$

Here P(0/-1) is the initial covariance matrix and is chosen such that $P(0/-1) \geq 0$. The sequence $\{e(t):=y(t)-H\hat{z}(t/t-1)\}$ is the innovations sequence. The equations (31) and (32) give an one-step-ahead estimate of the state, $\hat{z}(t+1/t)$, from information available upto and including time t. Further future predictions of the state and the output can be similarly obtained as discussed in section 6.4.

Remarks

1) If the matrices, Φ, Λ, Γ and H are constant then the steady state Kalman filter predictor can be used, which gives the same asymptotic error covariance of the estimated state vector as the time-varying, optimal predictor. The steady state Kalman gain, $\bar{K}$, is calculated using

$$\bar{K} := \Phi\bar{P}H^T(H\bar{P}H^T+R_v)^{-1} \qquad (35)$$

where $\bar{P}$ is the steady state value of the covariance matrix and is computed by solving the algebraic Riccati equation:

$$\bar{P} = \Phi\bar{P}\Phi^T - \Phi\bar{P}H^T(H\bar{P}H^T+R_v)^{-1}H\bar{P}\Phi^T + R_w \qquad (36)$$

Note that $\bar{K}$ and $\bar{P}$ can be computed *a priori* and thereby reduce computation.

2) In the adaptive case, Φ, Λ and Γ are time-varying as they use the estimated A, B and F polynomials defined in equation (18), and hence the Kalman gain has to be computed on-line. The numerical properties of the Kalman filter predictor can be improved using the scaling procedure discussed in Chapter 5 for ILS by minimizing $C\{P_s(t/t-1)\}$ where $P_s(t/t-1):=SP(t/t-1)S^T$ is the scaled covariance matrix and S is a diagonal and possibly time-varying scaling matrix. The equations (33) and (34) then become:

$$K(t) = \Phi S^{-1}P_s(t/t-1)S^{-T}H^T(HS^{-1}P_s(t/t-1)S^{-T}H^T+R_v)^{-1} \tag{37}$$

and

$$P_s(t+1/t)=\Phi S^{-1}P_s(t/t-1)S^{-T}\Phi^T-K(t)HS^{-1}P_s(t/t-1)S^{-T}\Phi^T + R_w \tag{38}$$

Since S is diagonal, its inverse is easily computed. Because $C\{P_s(t+1/t)\}$ is minimized equations (37) and (38) are numerically more robust for computer implementation than equations (33) and (34). Alternatively a square-root form of the predictor can be used with or without scaling. The square-root method only ensures that $P(t+1/t)\geq 0$ $\forall t$ where as using scaling additionally minimizes $C\{P_s(t+1/t)\}$.

3) For some applications the load-disturbance internal model F is approximated as $F(q^{-1})=\Delta=(1-q^{-1})$. This means that the load-disturbances are approximated as a sequence of random steps at random times or as a random-walk, i.e. $d(t)=e(t)/\Delta$. The reconstructed disturbance is then simply a summation of the residuals; the rate of summation or integration can be controlled by using $F(q^{-1})=k_I\Delta$ where k_I is a user specified constant. Note that this simplification makes the scheme equivalent to using a CARIMA model, but the output prediction is still done using positional or absolute values of $\{y(t)\}$ and $\{u(t)\}$ and hence does not suffer from the noise amplification problem.

6.3 Model identification and parameter estimation

The overall process model is composed of a model of the process dynamics and a model of the disturbances. For adaptive control, the parameters of these models are estimated on-line from the measured data, {y(t)} and {u(t)}. Such a model identification step forms an important part of the proposed adaptive control scheme and is shown as the parameter estimation block in Figure 6.1. The improved least squares algorithm is used in the parameter estimation as opposed to standard RLS.

The identification step involves estimating the parameter of two models: the first model represents the process dynamics and is simply called the process model. Its identification involves the estimation of the parameters of A and B polynomials and optionally C. The second model is the residual model whose identification involves fitting a single series (auto-regressive) model to the residual sequence {x(t)}.

6.3.1 Estimation of the process model parameters

The parameters of the process model are estimated from equation (19) which for use with the ILS algorithm is written in terms of zero-mean deviations of {y(t)} and {u(t)} as:

$$A(q^{-1})(y(t)-\bar{y}) = q^{-k-1}B(q^{-1})(u(t)-\bar{u}) + C(q^{-1})e(t) \tag{39}$$

C is assumed to be 1 for simplicity and ILS can be directly used. If $C \neq 1$ then extended least squares is used with the improvements suggested in Chapter 5 to obtain unbiased estimates of $\hat{A}(t)$ and $\hat{B}(t)$ (the estimate of C produced by ELS is not used directly in MAPC). Another alternative when $C\neq 1$ is to use a fixed noise polynomial T and filter the data using the low-pass filter $1/T(q^{-1})$ to eliminate the noise. Since ILS uses low-pass filtering of data the T filter can be included in this step. The estimation model then becomes:

$$A(q^{-1})\{y(t)/T(q^{-1}) - \bar{y}\} = q^{-k-1}B(q^{-1})\{u(t)/T(q^{-1}) - \bar{u}\} + C(q^{-1})e(t)/T(q^{-1}) \qquad (40)$$

If $T(q^{-1})=C(q^{-1})$ then only $e(t)$ appears in the above equation, which means unbiased estimates of A and B can be obtained. The above equation can be written as:

$$A(q^{-1})(y_f(t) - \bar{y}) = q^{-k-1}B(q^{-1})(u_f(t) - \bar{u}) + \eta(t) \qquad (41)$$

where $\eta(t)$ is a perturbation term. Note that the emphasis in this step is on obtaining unbiased estimates, $\hat{A}(t)$ and $\hat{B}(t)$ rather than on estimating the noise structure. Hence the use of a fixed, user chosen approximation for C. The noise is handled in the Kalman filter predictor via the residual model and by adjusting the noise covariances. For application of ILS equation (39) is written down in the following regression form:

$$y(t) = \theta_*^T\phi(t) + \eta(t) \qquad (42)$$

where $\theta_*:=(a_1, \ldots, a_n, b_1, \ldots, b_n)^T$ is the process parameter vector to be estimated; $\phi(t):=(\tilde{y}(t-1), \ldots,$

$\tilde{y}(t-n)$, $\tilde{u}(t-k-1)$, ..., $\tilde{u}(t-k-n))^T$ with $\tilde{y}(\cdot):=(y_f(\cdot)-\bar{y})$; $\tilde{u}(\cdot):=(u_f(\cdot)-\bar{u})$; $y_f(\cdot):=y(\cdot)/T(q^{-1})$ and $u_f(\cdot):=u(\cdot)/T(q^{-1})$. See Chapter 5 for the actual steps in ILS parameter estimation.

6.3.2 Estimation of the residual model parameters

The residuals defined by

$$\hat{\chi}(t) := \hat{A}(t)y(t) - q^{-k-1}\hat{B}(t)u(t) \tag{43}$$

contain the unmodelled process response such as noise, load-disturbances and unmodelled dynamics. $\hat{A}(t)$ and $\hat{B}(t)$ are found from the process model parameters, $\hat{\theta}(t)$, estimated at time t. Since the disturbances are considered additive at the output, for identifying the residual model the following residuals are used rather than those defined by equation (43):

$$\hat{\chi}'(t) := \frac{1}{\hat{A}(t)}\hat{\chi}(t) = y(t) - q^{-k-1}\frac{\hat{B}(t)}{\hat{A}(t)}u(t) \tag{44}$$

A single series model is then fit to the sequence $\{\hat{\chi}'(t)\}$ for use in the disturbance part of the state-space model. The parameters of this model are denoted by $\hat{\theta}_r(t)$. From section 6.2 it is known that (cf. equation (17) and (18)):

$$\hat{\chi}'(t) = \frac{C'(q^{-1})}{A(q^{-1})F(q^{-1})}e(t) = \frac{C(q^{-1})}{A(q^{-1})}e(t) + \frac{E(q^{-1})}{F(q^{-1})}e(t) \tag{45}$$

so that $\hat{\chi}'(t)$ contains noise as well as load-disturbances. It must therefore be noted that noise is modelled in the resulting state-space model in two ways: 1) noise explicitly added to the states and the output and which is optimally rejected by the Kalman filter predictor with an appropriate choice of the noise covariances; and 2) noise included in the residual model and which is predicted for use in the predictive controller. The net effect in the Kalman filter predictor is that the measurement noise is rejected whereas structured process noise is predicted for minimum variance control. Note that the residuals contain all unmodelled process effects and cannot be separated. For parameter estimation the following equation is used:

$$\hat{x}'(t) = \frac{1}{F(q^{-1})}e(t) = \frac{1}{\Delta D(q^{-1})}e(t) \tag{46}$$

where F can be considered to include C' and A. Since $\{\hat{x}'(t)\}$ may be non-stationary due to the presence of load-disturbances F is assumed to be given by $F(q^{-1}) := \Delta D(q^{-1})$ with D stable and equal to:

$$D(q^{-1}) := 1 + \sum_{i=1}^{n_r-1} d_i q^{-i} \tag{47}$$

The residual model parameters are therefore estimated using the regressor equation:

$$\Delta\hat{x}'(t) = \hat{\theta}_r(t)^T \phi_r(t) + e(t) \tag{48}$$

where $\hat{\theta}_r(t) := (\hat{d}_1(t), \ldots, \hat{d}_{n_r-1}(t))^T$; and $\phi_r(t) := (-\Delta\hat{x}'(t), \ldots, -\Delta\hat{x}'(t-n_r))^T$. Then $\hat{F}(t)$ is found from $\hat{D}(t)$ for use in the state-space model (i.e. equations (27) and (28)). Note that $\hat{\theta}_r(t)$ is also estimated using ILS.

Remark

There is a possibility that the identification of the two models may interact with each other, especially during closed-loop operation. Therefore it is necessary to delineate conditions under which each model can be estimated. This is discussed in section 6.7.1. Note that this two-stage identification can be compared to GLS (Clarke (1967)).

6.4 Output prediction

The MAPC uses an output prediction block as shown in Figure 6.1. Filtered ($\hat{z}(t/t)$) or one-step-ahead ($z(t+1/t)$) estimates of the state vector can be obtained using a Kalman filter. But it is also possible to obtain $\hat{z}(t+i/t)$ and hence $\hat{y}(t+i/t)$ for $i > 1$ using the Kalman filter, from measurements available up to and including time t. This particular estimator is referred to in this work as the multi-step-ahead Kalman filter predictor. It uses the one-step-ahead estimate, $\hat{z}(t+1/t)$, discussed in section 6.2.1 to give minimum variance estimates of the future states and is given by (Godfrey and Jones (1986)):

$$\hat{z}(t+i/t) = \Phi^{i}\hat{z}(t/t-1) + \Phi^{i-1}K(t)e(t) + \sum_{j=t}^{\bar{t}} \Phi^{\bar{t}-j}\Lambda u(j) \qquad (49)$$

and

$$\begin{aligned}\hat{y}(t+i/t) &= H\hat{z}(t+i/t)\\ &= H\Phi^{i}\hat{z}(t/t-1) + H\Phi^{i-1}K(t)e(t) + \sum_{j=t}^{\bar{t}} \Phi^{\bar{t}-j}\Lambda u(j) \qquad (50)\end{aligned}$$

for $i \geq 1$ and where $\bar{t}=t+i-1$. Since the state coefficient matrix, Φ, is block-diagonal by definition, just like in the one-step-ahead Kalman filter predictor, a lot of computation can be saved by considering two separate sub-systems: one for process dynamics and the other for disturbances. The overall process output can then be obtained by summing the outputs of each sub-system, i.e.,

$$\hat{x}(t+i/t) = \Phi_1{}^i\hat{x}(t/t-1) + \Phi_1{}^{i-1}K_1(t)e(t) + \sum_{j=t}^{\bar{t}} \Phi_1{}^{\bar{t}-j}\Lambda_1 u(j) \tag{51}$$

$$\hat{\xi}(t+i/t) = \Phi_2{}^i\hat{\xi}(t/t-1) + \Phi_2{}^{i-1}K_2(t)e(t) \tag{52}$$

$$\hat{y}(t+i/t) = y_m(t+i/t) + \hat{d}(t+i/t) \tag{53}$$

where $y_m(t+i/t)=H_1\hat{x}(t+i/t)$; $\hat{d}(t+i/t)=H_2\xi(t+i/t)$; and $K(t)=(K_1(t)^T,\ K_2(t)^T)^T$. With a little algebra one can write down the following equation using vector-matrix notation from equation (53):

$$\begin{aligned}\{\hat{y}(t+i/t), i\epsilon[1,N]\} = {} & \{H\Phi^i\hat{z}(t/t-1)+H\Phi^{i-1}K(t)e(t), i\epsilon[1,N]\} \\ & + A'\{u(t+i-1), i\epsilon[1,N]\}\end{aligned} \tag{54}$$

where $\{x(i), i\epsilon[1,N]\}$ denotes the vector $(x(1)\ \ldots\ x(N))^T$. Here A' is a lower triangular matrix with $a_{ij}' := H\Phi^{\bar{t}-j}\Lambda$ for $j \leq i$ and $a_{ij}' := 0$ otherwise. If the process has k sampling intervals of pure dead-time then the first k rows of A' will be zero. Equation (52) can be written in terms of the increments of u as follows:

$$\{\hat{y}(t+i/t), i\epsilon[1,N]\} = \{y^{*}(t+i/t), i\epsilon[1,N]\} + A'\{\Delta u(t+i-1), i\epsilon[1,N]\} \qquad (55)$$

where

$$\{y^{*}(t+i/t), i\epsilon[1,N]\} := \{H\Phi^{i}\hat{z}(t/t-1)+H\Phi^{i-1}K(t)e(t), i\epsilon[1,N]\} + \{(\sum_{j=1}^{i} a_{ij}')u(t-1), i\epsilon[1,N]\} \qquad (56)$$

Alternatively one can write:

$$\{y^{*}(t+i/t), i\epsilon[1,N]\}=\{y_{m}^{*}(t+i/t), i\epsilon[1,N]\}+\{\hat{d}(t+i/t), i\epsilon[1,N]\} \qquad (57)$$

where $y_{m}^{*}(t+i/t):=H\Phi_{1}^{i}\hat{x}(t/t-1)+H\Phi_{1}^{i-1}K_{1}(t)e(t)+(\sum_{j=1}^{i} a_{ij}')u(t-1)$ is the process model based open-loop prediction of the process response (cf. feedforward path) using measured signals, and $\hat{d}(t+i/t)$ is the prediction of the unmodelled process response (cf. feedback path). Thus from equation (55) one can see that the future predictions of the process outputs, $\{\hat{y}(t+i), i\epsilon[1,N]\}$, consist of two parts: $\{y^{*}(t+i/t), i\epsilon[1,N]\}$ calculated from past inputs plus past and present outputs; and $A'\{\Delta(t+i-1), i\epsilon[1,N]\}$ due to present and future control actions.

Remarks

1) The proposed control scheme uses a control horizon, N_u, beyond which the control increments are set to zero, i.e. $\Delta u(t+i-1)=0$ for $N \geq i > N_u$. This means that $u(t+i-1)=u(t+N_u-1)$ for $i>N_u$ and the A' matrix in equation (54) is replaced by $A:=A'S$ so that equation

(55) becomes:

$$\{\hat{y}(t+i/t), i\epsilon[1,N]\} = \{y^*(t+i/t), i\epsilon[1,N]\} + A\{\Delta u(t+i-1), i\epsilon[1,N_u]\} \tag{58}$$

where S is of the dimension NxN_u and is given by

$$S = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 1 & 1 & 0 & \dots & 0 \\ . & . & . & \dots & . \\ . & . & . & \dots & . \\ 1 & 1 & 1 & \dots & 1 \\ . & . & . & \dots & . \\ 1 & 1 & 1 & \dots & 1 \end{bmatrix}$$

2) Note that the output prediction does not require solving a single or a bank of diophantine identities, e.g. as in GPC; instead the prediction requires calculation of the Kalman filter predictor gain, K(t), and Φ^i, $i\epsilon[1,N]$ which are the major computational steps. However, actual computation of Φ^i is simplified by noting the fact that it is block-diagonal and consists of sparse matrices with special (canonical) structure. Furthermore, if the state-space model is time-invariant then the steady state Kalman filter predictor can be used with $K(t):=\bar{K}$ computed *a priori*. If the state-space model is time-varying (cf. self-tuning case), then the parameters of the state-space model at time (t-1) are updated at time t using $\hat{\theta}(t)$ and $\hat{\theta}_r(t)$. The Kalman filter calculations (i.e. computation of P(t+1/t) and K(t)) are done as if the state-space model has not changed.

3) If an auxiliary measured input, r, is available then it can be treated just like the control input in the predictor and can be compensated in a feedfoward manner. Since the future values of the measured input, $r(t+i)$, are unkown, they are set to the current value, i.e. $r(t+i)=r(t)$, $i>1$. This means the measured input is assumed not to change over the time interval $(t+1)$ to $(t+N)$.

The details of the Kalman filter predictor are schematically shown in Figures 6.2 and 6.3 for the time-invariant case and in Figures 6.4 and 6.5 for the self-tuning or time-varying case. Note that the feedforward path and feedback path are indicated by dashed lines in Figure 6.2.

6.5 The multi-step predictive controller

The multi-step predictive controller shown in Figure 6.1 generates a trajectory of future control actions such that the predicted output trajectory produced by the multi-step-ahead Kalman filter prediction is as close to the reference trajectory as possible (in the sense that a user specified cost function is minimized). The supervisory system, also shown in Figure 6.1, supplies the reference trajectory, $\{y_{ref}(t+i), i\in[1,N]\}$, at time t. The reference trajectory may be a fixed or filtered setpoint, computed by some other means or simply specified *a priori*. The control strategy is based on the minimization of the following multi-step, weighted, quadratic cost function which is similar to the cost functions used in GPC, DMC, MOCCA etc.,

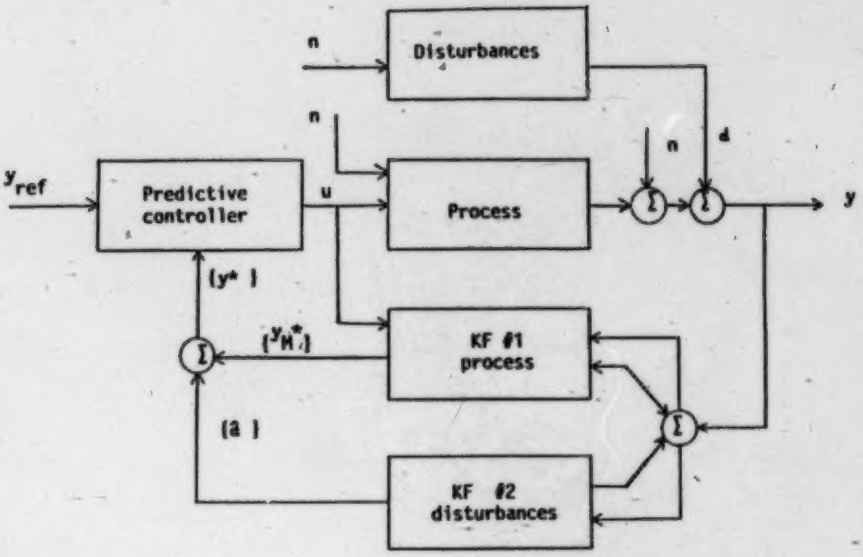

n
Disturbances
n
n
d
y_{ref}
Predictive controller
u
Process
y
{y* }
$\{y_M^*\}$
KF #1 process
{a }
KF #2 disturbances

Figure 6.2 A high-level block diagram of non-adaptive MAPC

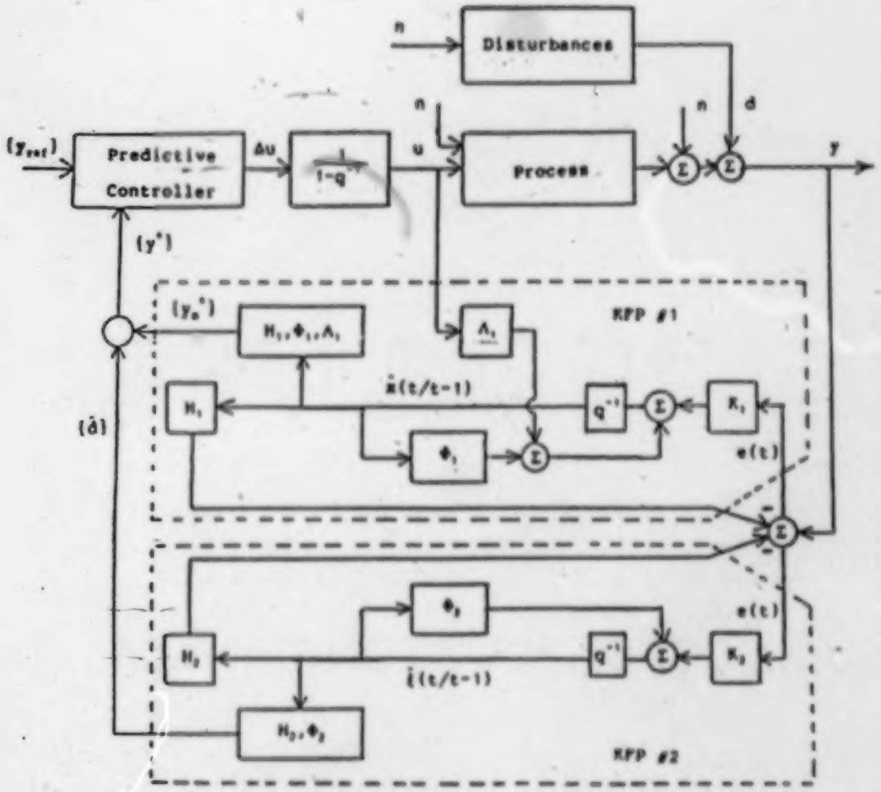

n
Disturbances
n
n
d
{y_ref}
Predictive Controller
Δu
1/(1-q⁻¹)
u
Process
Σ
Σ
y
{y*}
{y_m*}
H₁, Φ₁, Λ₁
Λ₁
KFP #1
x̂(t/t-1)
H₁
q⁻¹
Σ
K₁
{d̂}
Φ₁
Σ
e(t)
Σ
Φ₂
e(t)
H₂
q⁻¹
Σ
K₂
ξ̂(t/t-1)
H₂, Φ₂
KFP #2

Figure 6.3 A detailed block diagram of non-adaptive MAPC

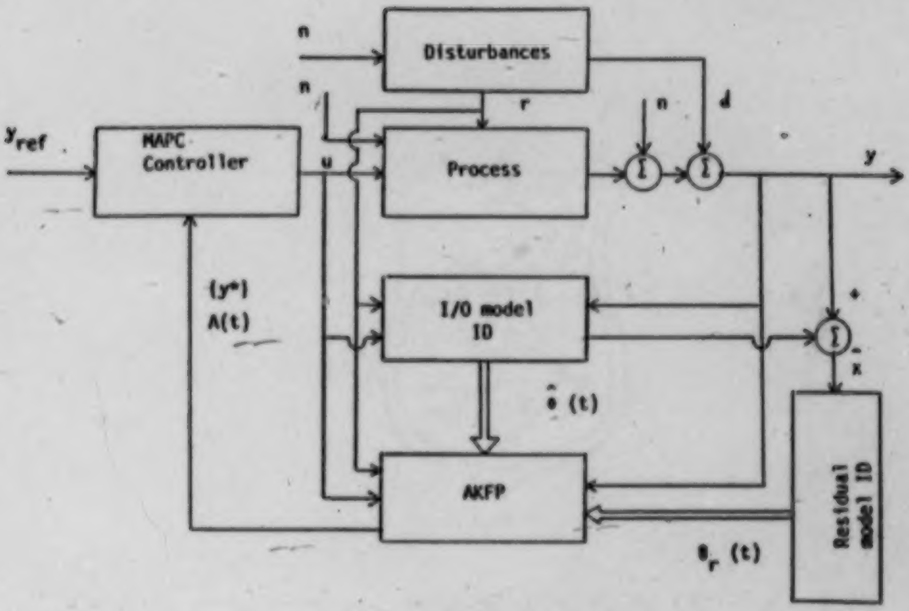

n
Disturbances
n
r
n
d
y_ref
MAPC Controller
u
Process
Σ
Σ
y
{y*}
A(t)
I/O model ID
+
Σ
x
θ̂ (t)
AKFP
θ_r (t)
Residual model ID

Figure 6.4 A high-level block diagram of MAPC

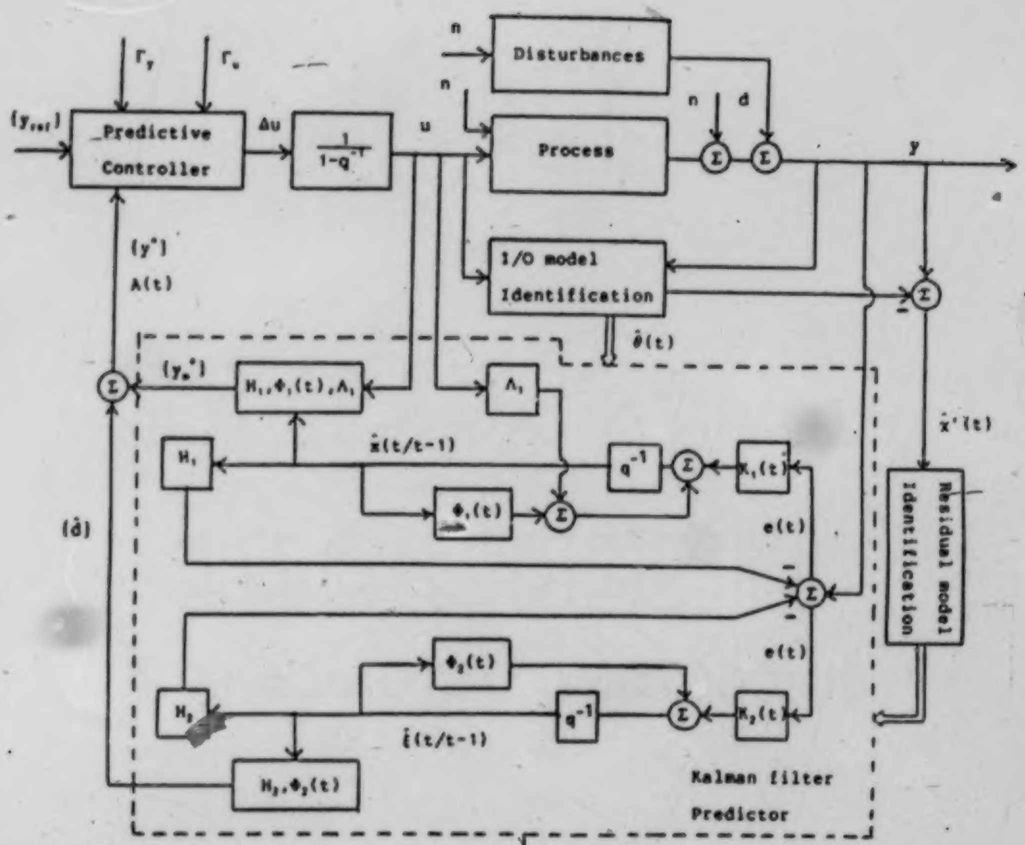

n
Disturbances
n
Γ_y
Γ_u
{y_ref}
Predictive Controller
Δu
1/(1-q^-1)
u
Process
n
d
Σ
Σ
y
I/O model Identification
Σ
{y*}
A(t)
θ̂(t)
{y_m*}
H_1,Φ_1(t),Λ_1
Λ_1
Σ
H_1
x̂(t/t-1)
q^-1
Σ
K_1(t)
x̂'(t)
Φ_1(t)
Σ
e(t)
{d̂}
Σ
Residual model Identification
e(t)
Φ_2(t)
H_2
q^-1
Σ
K_2(t)
ξ̂(t/t-1)
H_2,Φ_2(t)
Kalman filter
Predictor

Figure 6.5 A detailed block diagram of MAPC

$$J(N_1,N_2,N_u) = \frac{1}{2}\Big[\sum_{i=N_1}^{N_2} \{y_{ref}(t+i)-\hat{y}(t+i/t)\}^2\gamma_{yi}(t) + \sum_{i=1}^{N_u} \{\Delta u(t+i-1)\}^2\gamma_{ui}(t)\Big] \qquad (59)$$

where $\{\hat{y}(t+i/t), i\in[N_1,N_2]\}$ are the predictions of the future process outputs from measurements available up to and including time t. Note that the Kalman filter predictor generates $\{\hat{y}(t+i/t), i\in[1,N_2]\}$ so that $N=N_2$ in section 6.4. Here $N_2>N_1\geq 1$; $N_y:=N_2-N_1+1$ is a finite output horizon; N_u is a finite control horizon with $N_u\leq N_y$; and $\{\gamma_{yi}\}$ and $\{\gamma_{ui}\}$ are non-negative and possibly time-varying weights on the tracking error and control action respectively.

Substituting for $\{\hat{y}(t+i/t), i\in[N_1,N_2]\}$ using equation (58) and defining $\Gamma_y:=\text{diag}(\gamma_{yi})$, and $\Gamma_u:=\text{diag}(\gamma_{ui})$ one gets:

$$\begin{aligned} J(N_1,N_2,N_u) := \frac{1}{2}\Big[&(\{y_{ref}(t+i), i\in[N_1,N_2]\}-\{y^*(t+i/t), i\in[N_1,N_2]\} - \\ &A\{\Delta u(t+i-1), i\in[1,N_u]\})^T\Gamma_y(\{y_{ref}(t+i), i\in[N_1,N_2]\} - \\ &\{y^*(t+i/t), i\in[N_1,N_2]\} - A\{\Delta u(t+i-1), i\in[1,N_u]\}) + \\ &\{\Delta u(t+i-1), i\in[1,N_u]\}^T\Gamma_u\{\Delta u(t+i-1), i\in[1,N_u]\}\Big] \end{aligned} \qquad (60)$$

where A is of the form:

$$A := \begin{bmatrix} a_1 & 0 & \cdots\cdots & 0 \\ a_2 & a_1 & \cdots\cdots & 0 \\ \cdot\cdot & \cdot\cdot & \cdots\cdots & \cdot \\ \cdot\cdot & \cdot\cdot & \cdots\cdots & \cdot \\ a_{N_u} & a_{N_u-1} & \cdots\cdots & a_1 \\ \cdot\cdot & \cdot\cdot & \cdots\cdots & \cdot \\ a_{N_y} & \cdot\cdot & \cdots\cdots & a_{N_y-N_u+1} \end{bmatrix}$$

and is found by taking the rows N_1 to N_2 of the vector-matrix equation (58).

Remarks

1) If $N_1=(k+1)$ then $a_i>0$ for $i=1, 2, \ldots$ and the A matrix is non-singular.

2) $\{a_1, a_2, \ldots\}$ defines the unit step response of the process.

The control-law is obtained by setting $\frac{\partial J}{\partial\{\Delta u(t+i-1), i\in[1,N_u]\}}=0$, which gives:

$$\{\Delta u(t+i-1), i\in[1,N_u]\}=(A^T\Gamma_y A+\Gamma_u)^{-1}A^T\Gamma_y\{y_{ref}(t+i)-y^*(t+i/t), i\in[N_1,N_2]\}$$

$$= A^* \{y_{ref}(t+i)-y^*(t+i/t), i\in[N_1,N_2]\} \qquad (61)$$

where

$$A^*:=(A^T\Gamma_y A+\Gamma_u)^{-1}A^T\Gamma_y \qquad (62)$$

A high-level block diagram of the basic, non-adaptive controller is presented in Figure 6.2. This diagram is redrawn in Figure 6.3 with further details added to the Kalman filter prediction part of the diagram.

Remarks

1) Control offset usually results when absolute values of the control signal, u, are used in the cost function, J (cf. equation (59)). Therefore increments of u(·), Δu(·), are penalized which then introduces integral action into the controller.

2) The numerical robustness of the matrix inversion in computing A^* by equation (62) can be significantly improved by using the optimal scaling discussed in Chapter 5. Essentially this involves inverting a scaled $(A^T\Gamma_y A+\Gamma_u)$ rather than inverting it directly. A diagonal scaling matrix, S, can be computed as in Chapter 5 (cf. Theorem 3) to minimize $C\{S(A^T\Gamma_y A+\Gamma_u)S^T\}$. Then A^* is computed from

$$A^* = S^T[S(A^T\Gamma_y A+\Gamma_u)S^T]^{-1}S^{-1}A^T\Gamma_y \qquad (63)$$

which is significantly more robust numerically than equation (60).

The control-law given by equation (61) involves very little computation when the process model parameters are fixed, since A^* can then be computed *a priori*. In the self-tuning case, since the parameters change on-line, A^* needs to be recalculated at every time instant. Note that the calculation of A^* involves a matrix inversion which is a computationally significant step. The dimension of A is $N_y x N_u$ and A^* is $N_u x N_y$ but the dimension of the matrix to be inverted is only $N_u x N_u$. Because of the moving-window formulation it is generally possible to choose N_y as low as 3 or 5 and N_u as low as 1 or 2. Thus A is usually a small matrix of the order of 5x2 and the matrix to be inverted is only of the order of 2x2.

The control-law is usually implemented in a receding-horizon sense, i.e. only the first control action, $\Delta u(t)$, is implemented and the calculations are repeated at the next sampling interval. This means that only the product of the first row, g^T, of A^* and $\{y_{ref}(t+i)-y^*(t+i/t), i\epsilon[N_1,N_2]\}$ needs to be computed,

$$\text{i.e. } \Delta u(t) = g^T\{y_{ref}(t+i)-y^*(t+i/t),\ i\epsilon[N_1,\ N_2]\} \qquad (64)$$

Remarks

1) The receding-horizon approach makes the control algorithm time-invariant, i.e. the same control-law (cf. equation (64)) is used at each time instant, unlike in a fixed-horizon LQG policy.

2) When the complete reference trajectory is known ahead of time and when the process model is relatively time-invariant with good parameters it is some times possible to implement the whole control vector, $\{\Delta u(t+i-1), i\epsilon[1,N_u]\}$, over the control horizon of N_u and repeat the calculations every N_u intervals. This saves some computation.

3) The control scheme is made self-tuning by using the estimated process and residual models (i.e. $\hat{\theta}(t)$ and $\hat{\theta}_r(t)$ respectively) at each time in the Kalman filter predictor and predictive controller blocks. Figures 6.4 and 6.5 show the details of the adaptive controller schematically.

4) It is common practice in control system design to include a model of the external disturbances in the controller using the internal model principle to achieve proper disturbance rejection (cf. integral action to get rid of step load-disturbances). In the present scheme the controller is essentially an open-loop inverse model controller and therefore the disturbances are eliminated via the feedback path using the same inverse model controller. This is equivalent to eliminating disturbances using feedforward control based on a predicted (forecast) disturbance or residual. However, for good prediction of the disturbance a proper internal model of the disturbance or a good on-line estimated residual model is required.

6.5.1 Choice of the controller parameters

The parameters N_1, N_2, N_y and the weighting matrices Γ_y and Γ_u allow one to shape the dynamic response of the controlled process as well as to stabilize NMP processes or processes with open-loop unstable or poorly damped dynamics. If the output horizon and the control horizon are equal, the control problem is essentially solving a set of linear algebraic equations for the control vector. By choosing the control horizon less than the output horizon in the problem formulation one then has to solve an over-determined set of linear algebraic equations for the control vector. The proposed method solves this problem using the weighted least squares approach. Thus by assuming that the control increments beyond the control horizon to be zero instead of allowing them to be free, the method derives a lot of flexibility as well as reduces computation. In particular, this method allows one to stabilize NMP processes. Choosing $\Delta u(t+i-1)=0$ for $i>N_u$ can also be interpreted as using ∞-weighting on $\{\Delta u(t+i-1), i\epsilon[N_u+1, N_y]\}$ in the cost function.

Output horizon (N_1, N_2)

It has already been pointed out that using prediction over a range of points in the controller cost function rather than a single point makes the controller robust to incorrect specification of the process dead-time. If the time-delay, k, of the process is exactly known then one can take $N_1=(k+1)$, since in that case the first future output that is directly affected by the current control action is at $t+k+1$. Thus using $\hat{y}(t+1/t)$ to $\hat{y}(t+k/t)$ in the cost function is redundant and hence by omitting these some computation can be saved. If the time-delay is expected to vary or is not known with certainty then one can set $N_1=1$. In that case, even though A is singular a solution can be forced to exist by a proper choice of the weighting matrices (e.g. choose $\Gamma_u=I$). This is possible because of the over-determined set of equations for the solution of the control vector. Thus the proposed controller can perform well even if the knowledge of the process time-delay is poor. Single-step cost functions do not offer this flexibility and controllers based on these may fail or perform poorly if the time-delay is not exactly known. For NMP processes, N_1 can be set beyond the initial wrong way response in the process step response so that only the points that approach the setpoint in the right direction are included in the controller cost function.

N_2 can be chosen so that $N_y \cong 5$ to 10. Alternatively the output horizon, N_y, can be chosen equal to the rise time of the process step response and N_2 is then set appropriately. This is adequate in most cases. If the process is open-loop unstable or has complex poles, then obviously more points must be included in the output horizon. One approach for selection of N_y is through a simulation study using a rough model of the process.

As a default N_1 can be set to 1 and N_2 can be set to 10. Most processes give stable response and good performance with these settings. If more conservative performance is desired, N_2 can be increased to cover the entire open-loop step response.

Control horizon (N_u)

Usually there is no advantage in taking $N_u > n$ where n is the order of the process. If there is no weighting then for $N_u \geq n$ the controller gives output dead-beat control. In most cases N_u can be set to 1 or 2.

Weighting matrices (Γ_y, Γ_u)

The weighting matrices (Γ_y and Γ_u) are especially useful in the multivariable case because they can be used to specify that any particular output (input) is more or less important than the other outputs (inputs) (i.e. relative weighting among outputs and or inputs). The weighting matrices can also be used to penalize the control action or to ensure that soft constraints on the process variables are not violated. In numerical terms, the weighting matrices are very helpful to force a stable solution to the control problem. For example, when A is poorly conditioned or almost singular then in the absence of any weighting equation (61) can give poor solutions. This may result in excessive oscillations or ringing in the control input or even closed-loop instability. By choosing the weighting matrices appropriately the matrix inversion in equation (62) can be done more robustly (i.e. $C\{A^T\Gamma_y A+\Gamma_u\}$ can be significantly reduced). This can be done from a knowledge of the process time scale (i.e. sampling interval), a rough step response of the process and the desired closed-loop performance. In the SISO case it is simpler to choose $\Gamma_y=I$ and $\Gamma_u=\lambda I$, $\lambda>0$, so that

$$A^* = (A^T A+\lambda I)^{-1} A^T \tag{65}$$

in equation (61). There is only one tuning parameter, λ, to be adjusted on-line. If $\lambda=0$ then there is no weighting on the control action. If λ is large then the control action is heavily penalized.

6.5.2 Special cases of the proposed controller

The proposed controller (cf. equation (61)) reduces to several special cases which are pointed out in this section. As these special cases are simple and widely used they help to understand the proposed scheme.

Minimum variance regulator

When $N_1=N_2=(k+1)$, $N_u=1$, $\gamma_y=1$ and $\gamma_u=0$ the control-law given by equation (61) reduces to the one-step-ahead (output dead-beat) controller. Since the Kalman filter predictor generates minimum variance estimates of the output the result is a minimum variance regulator. Thus Astrom's minimum variance regulator (Astrom (1970)) is a special case of the proposed scheme, but the latter is more general due to the incorporation of an internal model of the load-disturbances. Aström's minimum variance regulator is based on the assumption that the noise structure is characterized by a noise observer polynomial, C, in a CARMA model and the predictor is generated by solving a diophantine identity that uses C. The Kalman filter predictor on the other hand uses the noise covariances and the residual model to generate minimum variance predictions. The two approaches can be made equivalent if one notes that C can be derived from the innovations state-space model generated by the Kalman filter predictor,

$$\text{i.e. } C(q^{-1}) = 1 + (a_1+k_1(t))q^{-1} + \ldots + (a_n+k_n(t))q^{-n} \qquad (66)$$

where $K_1(t)=(k_1(t) \ldots k_n(t))^T$ is the Kalman gain vector (cf. section 6.2). However, the Kalman filter predictor is easier to tune on-line using the noise covariances which provide a direct characterization of the noise. The C polynomial is difficult to specify and to change when the noise characteristics of the process vary since C is difficult to estimate on-line.

Generalized minimum variance controller

The generalized minimum variance (GMV) controller of Clarke and Gawthrop (1979) can also be seen to be a special case of the proposed controller when $N_1=N_2=(k+1)$, and the control action is weighted (e.g. $\gamma_y=1$ and $\gamma_u>0$).

Extended horizon adaptive controller

If $N_1=N_2=d>(k+1)$ and $N_u=1$ with $\gamma_y=1$ and $\gamma_u=0$ the proposed controller becomes Ydstie's extended horizon adaptive controller (Ydstie (1984)). Ydstie's controller has been shown theoretically to be a stabilizing controller for a stable, NMP process. However, it has not been shown to stabilize open-loop unstable processes. Also for processes with poorly damped poles simulation experience shows that the extended horizon method is unstable, unlike the proposed scheme which uses more than one output point in the cost function. This is because when the process has complex poles more than one future output needs to be accounted for in the cost function.

Smith predictor

The Smith predictor (Smith (1957)) is widely used in process control for dead-time compensation. Its structure is shown in Figure 6.6. The proposed scheme can be compared to a Smith predictor when the predictive controller is replaced by an error driven feedback controller based on $\hat{y}(t+k+1/t)$ as shown in Figure 6.7. Since the Kalman filter effectively filters out high-frequency noise and unmodelled dynamics on the process output, it decreases the effect of these on the closed-loop and improves the robustness of the Smith predictor. Feedback control using Kalman filter prediction can therefore be interpreted as a Smith predictor with a low-pass filter as shown in Figure 6.8 (this fact was theoretically shown by Walgama (1986)). The low-pass filter can be a simple exponential (or first order) filter which filters out high-frequency components (such as noise) of the signal going to the controller. The Kalman filter predictor has the same filtering effect on the proposed multi-step predictive controller which improves its robustness (this has been verified in simulations, e.g. Example 5 in the

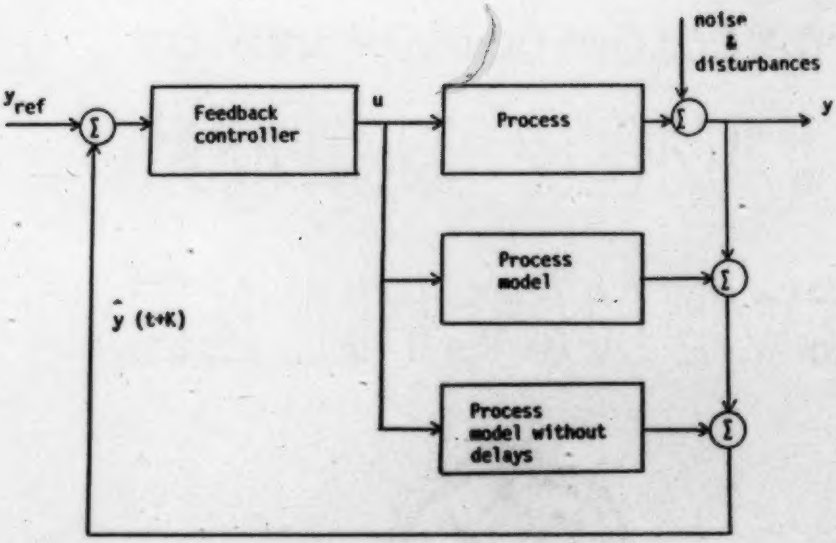

noise
&
disturbances
y_{ref}
Σ
Feedback controller
u
Process
Σ
y
Process model
Σ
$\hat{y}$ (t+K)
Process model without delays
Σ

Figure 6.6 Smith predictor

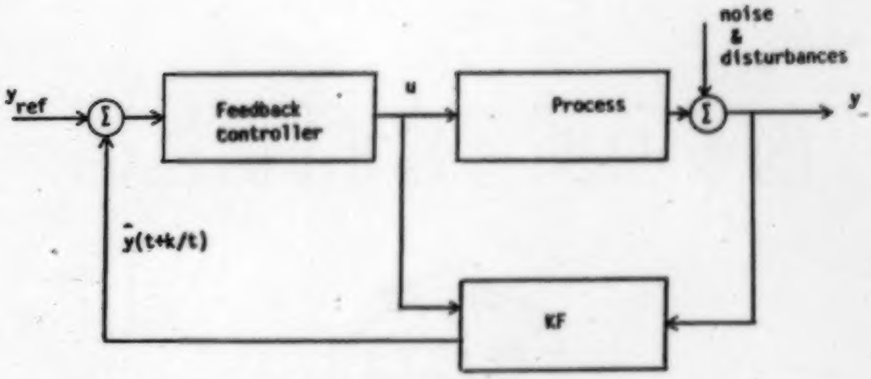

noise
&
disturbances
y_{ref}
Σ
Feedback controller
u
Process
Σ
y
$\hat{y}(t+k/t)$
KF

Figure 6.7 Kalman filter predictor based feedback controller

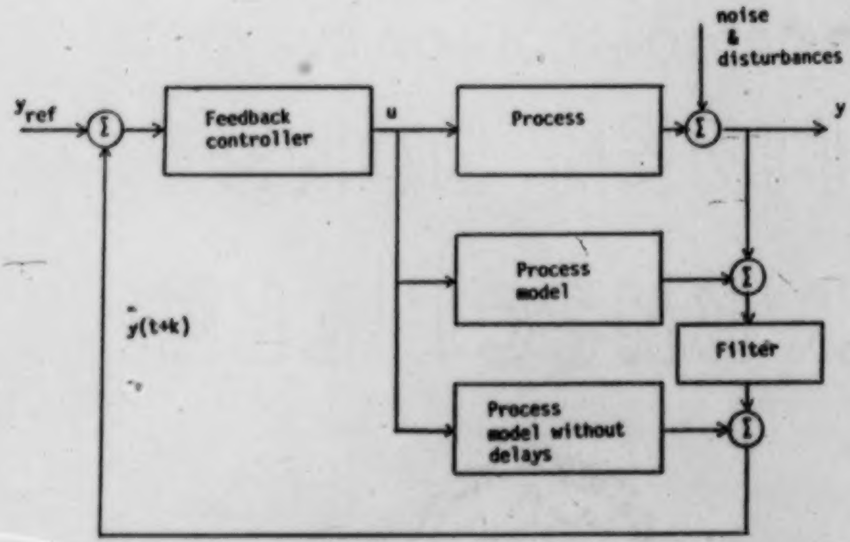

y_{ref}
Σ
Feedback controller
u
Process
noise & disturbances
Σ
y
Process model
Σ
Filter
Process model without delays
Σ
$\tilde{y}(t+k)$

Figure 6.8 Smith predictor with a filter

Appendix).

Internal model controller

The feedback controller using Kalman filter prediction can also be compared to the internal model controller (IMC) of Garcia and Morari (1982) which is shown in Figure 6.9. The IMC scheme uses a low-pass filter in its feedback path to reduce the effect of the model-process mismatch on the closed-loop. It has been shown theoretically that the filter in the feedback path can be used to improve the robustness of the controller (i.e. to trade off performance for robustness) and hence supports the intuitive observation made above.

6.6 The supervisory system

The MAPC assumes that the reference trajectory, $\{y_{ref}(t+i), i\epsilon[N_1,N_2]\}$ is available at time t. The supervisory system which is shown in Figure 6.1 generates these values. The reference input may be constant and equal to a setpoint over the output horizon or may vary. For example, the supervisory system can be used to generate a smoothed setpoint trajectory as in IDCOM (1978) where it is considered that a smoothed approach to the current setpoint, y_{sp} is required from the current output. In this case the supervisory system can calculate the reference trajectory from the following first order lag model:

$$y_{ref}(t) = y(t) \tag{67}$$

$$y_{ref}(t+i) = \alpha y_{ref}(t+i-1) + (1-\alpha)y_{sp}(t), \quad i=1,\ 2,\ \ldots,\ N_2 \tag{68}$$

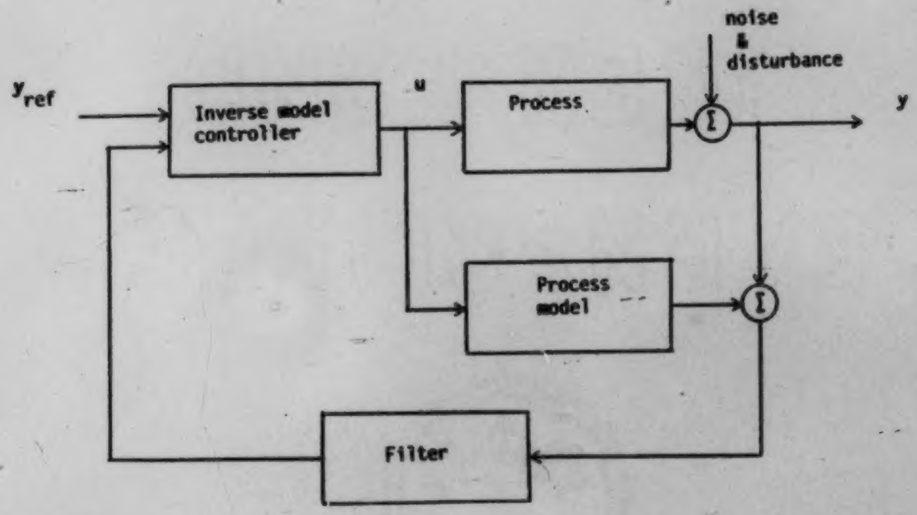

noise
&
disturbance
y_{ref}
Inverse model controller
u
Process
Σ
y
Process model
Σ
Filter

Figure 6.9 Internal model controller (IMC)

or equivalently $y_{ref}(t) = \frac{1-a}{1-aq^{-1}} y_{sp}(t)$ (69)

$a \cong 1$ for a smoothed transition from current output to the setpoint. MAPC is thus capable of both time-varying and time-invariant reference inputs. Another case is that the reference trajectory may be specified *a priori* by the process operator. This ability to follow prespecified reference trajectories is very useful in practical applications and there are a number of real examples where such an ability is needed. Some examples are robot motion and batch reactor control where the temperature of the batch may have to follow a specified trajectory.

Handling constraints

The reference inputs can also be adjusted on-line to satisfy process constraints. Constraints on process variables (e.g. u and y) are very important in practical applications and are usually ignored in the adaptive control literature. Amplitude constraints on the control signal, u, are especially important. Saturation of the control signal may appear in an adaptive scheme, e.g.

1) during start-up or after parameters have converged;
2) due to bad initialization;
3) due to unattainable desired performance (i.e. y_{ref}).

When saturation of the input signal occurs the process ceases to be linear and its effect is particularly deleterious on the parameter estimation, as the estimator then tends to identify a zero-gain process. The process can be forced to remain in in the linear region by imposing smoother reference changes or calculating the reference signal such that the limits on u are never violated. The supervisory system can be used to achieve this task.

The procedure is outlined for the SISO case. First the control signal is calculated using the control-law (i.e. equation (61)) assuming normal reference trajectory. If the calculated control signal satisfies the constraints over the control horizon, then the current control action is implemented. Otherwise a new reference trajectory $\{\hat{y}_{ref}(t+i), i \in [N_1, N_2]\}$ is calculated by solving the optimization problem:

$$\text{minimize (w.r.t. } \{\hat{y}_{ref}\}) \ \frac{1}{2}\|\{\hat{y}_{ref}\}-\{y_{ref}\}\|^2$$

$$\text{subject to } \{\Delta u\} = A^*\{\hat{y}_{ref}-y^*\}$$

$$u(t) \in [u^-, u^+] \text{ for all } t$$

where u^- and u^+ are the lower and upper limits on the control signal. This problem can be restated as:

$$\text{minimize (w.r.t. } \{\hat{y}_{ref}\}) \ \frac{1}{2}\|\{\hat{y}_{ref}\}-\{y_{ref}\}\|^2$$

$$\text{subject to } h^- \le A^*\{\hat{y}_{ref}\} \le h^+$$

where $h^- := u^- - u(t-1) + SA^*\{y^*\}$ and $h^+ := u^+ + u(t-1) + SA^*\{y^*\}$ with S as defined earlier in section 6.4.

The interpretation is that the new reference trajectory is calculated with minimum deviation from the original reference trajectory so that the control signal calculated by the predictive controller using the new reference trajectory satisfies the input amplitude constraints. The solution is iterative because of the inequality constraints.

Note that the computation can be significantly reduced if the reference trajectory is time-invariant and is equal to a constant setpoint. In this case if the calculated input signal, u(t), satisfies the constraints then it is implemented. Otherwise the calculations are repeated with a new setpoint, $\hat{y}_{sp}$, which is computed e.g. by $\hat{y}_{sp}=y_{sp}-\delta \mathrm{sign}(u(t))$ where δ is a small positive number. The procedure is repeated until u(t) lies in the linear region. At the next sampling interval or more appropriately after N_u intervals the control calculations can be resumed with the original setpoint or reference trajectory. Also note that the controller and supervisory blocks interact and the executive sub-system can be used to control the interaction as shown schematically in Figure 6.1.

The constraints can also be handled by the predictive controller directly. The advantage of using the supervisory system is that it admits more flexibility.

6.7 The AI executive

One of the requirements of a practical adaptive controller is that it should be easy to commission and operate. It is common in adaptive control practice to use heuristic fixes or safety jackets (e.g. in parameter estimation) so that the adaptive controller will work smoothly over a broader range of conditions. To do this it is usually suggested (Iserman and Lachmann (1985), Schumann *et al.* (1982) and Astrom (1983a)) that *another block be added to direct the overall operation of the adaptive controller as shown in Figure 6.1*. The functions of such a block in the proposed controller can be manifold, e.g.

1) to start-up or commission the adaptive controller;

2) to perform mode-switching:

start-up(non-adaptive) → adaptive;
emergency (adaptive) → sutdown/a backup, fixed parameter controller;

3) to supervise:

Model identification:
-switch on/off identification of process model;
-selection of the parameters (ι, κ, tr) in ILS for each of the two identification steps;
Kalman filter prediction:
-selection of the noise covariances (R_w, R_v);
-discounting the residual model (i.e. $\hat{\theta}_r(t)$);
MAPC:
-selection of the controller parameters N_y, N_u, Γ_y and Γ_u to obtain satisfactory performance.

The knowledge Based Systems (AI) provide a suitable framework to implement these functions. An example application of AI in real-time control of a simple process is discussed in Chapter 2. The discussion here is focused on the functional aspects of an AI based executive for MAPC rather than its complete design and implementation. Note that the development of the complete executive sub-system is not the objective of the work reported here. However, for demonstration purposes selected functions (e.g. supervision of the model identification step) are implemented and tested in a conventional programming framework. This work is discussed in the next section.

Start-up

To start the adaptive controller, initial estimates $\hat{\theta}(0)$ and $\hat{\theta}_r(0)$, and a knowledge of the delay k are needed. These can be generated in several ways, e.g. under closed-loop using an existing non-adaptive controller or in open-loop operation. Two methods: open-loop step response testing and Astrom's on/off regulator (Astrom (1986)) look very appealing. Any of these can be used to generate suitable estimates of k and $\hat{\theta}(0)$. The latter approach has the advantage that it can be applied with an existing feedback controller without upsetting normal operation too much. It is also safer. Arzen (1986) discusses how to incorporate the start-up procedure of Astrom in a rule based system.

Mode-switching

This function performs the task of switching bumplessly during start-up (from the existing controller to the adaptive controller) and during an emergency (from the adaptive controller to a back-up controller or to a shut-down procedure). A typical emergency situation is when the control system becomes unstable for some reason. This part of the executive sub-system has to recognise such emergency situations in real-time. A method for detecting unstable signals is presented in (Gertler and Chang (1986)) and may be used here.

Smooth or improved operation

The main objective of this part of the executive sub-system is to supervise the performance of the proposed adaptive controller. For example it is necessary to ensure that the parameter estimates, especially, $\hat{\theta}(t)$, are reasonable for all t; that the controller performance is satisfactory etc. Several rules of thumb can be readily identified to achieve these things. Some typical rules that are used to supervise the model identification step are discussed in the next section.

Note that the AI based executive sub-system has to make its decisions in real-time based on sensor-data (i.e. measurements), performance specifications etc. The structure and inference (decision) time of such a system can be very crucial. Some characteristics of such RTSB AI applications were summarized in Chapter 2.

6.7.1 Some example rules used in the supervision of the model identification step of MAPC

A two-stage identification scheme is used in MAPC to separately estimate the parameters of the process model and the disturbance model. The decomposition of the total process response into these two models is a key feature of the proposed control algorithm and is an alternative to the use of a single model, e.g. CARMA model. The two-stage scheme allows better estimation and prediction of disturbances. The disturbance model is estimated from a residual sequence which represents the unmodelled process response such as process noise, load-disturbances and unmodelled dynamics not included in the process model. Usually these various effects in the residuals cannot be distinguished in practice. These must however be included in the prediction of the process output for offset-free regulation. The stability of MAPC depends only on the process model. Therefore, the process model must represent the dominant dynamics of the process which can result in a stable and satisfactory closed-loop behaviour. The residual model is a fast, higher-order, single series (cf. AR type) model that represents the unmodelled response.

The identification of the process model should be done under carefully controlled conditions where as the residual model can be estimated all the time to track the fast variations in the residuals that are associated with disturbances. Note that the most important detriment to the process model is the occurrence of a load-disturbance which can generate strong excitation in the closed-loop. If the estimation of the process model is continued, its parameters will be corrupted. Although the load-disturbance may be eliminated, the resulting process model can cause poor or even unstable closed-loop behavior subsequently. Hence the estimation of the process model must be stopped when a load-disturbance occurs. The load-disturbance usually results in a shift in the mean-level of the residual sequence and can be detected easily. However, process parameter variations can also cause a shift in the mean-level of the residual sequence but in this case the estimation of the process model must be allowed to go on to track these variations. The advantage of the proposed scheme is that the unmodelled dynamics resulting from the parameter variations are temporarily included in the residual model and hence included in the predictive control. When estimation of the process model is turned-on again (e.g. when the setpoint changes) the parameter variations are picked up by the process model and the residual model is adjusted appropriately. The following informally stated rules can be used in the control of the identification step.

Identification of the process model

1) Switch the parameter estimation on if the setpoint changes (or the reference trajectory is time-varying).
2) Switch the parameter estimation on if external excitation is added to improve identification of the process model.
3) Switch the parameter estimation off if a load-disturbance is detected. The detection is done using a CUSUM test on the residual sequence and is discussed below. However, the estimation of the mean values of u and y must go on with suitable forgetting factors (λ_u and λ_y) in the ILS procedure to track the resulting change in the d.c. bias of the process model equation so that when the estimation of the process model is resumed the effect of the bias on the process model parameters is minimized.
4) Select the parameters in the ILS procedure. These parameters are discussed in Chapter 5 and simple rules can also be specified for their selection. Note that ILS has its own criteria (cf. ι and κ) for turning the parameter estimation off. But this is not the same as the rules specified above.

Identification of the residual model

1) The parameter estimation is on all the time except when the ILS procedure for this part of the identification turns off due to poor excitation (cf. ι and κ).

2) Discount the residual model if the prediction or forecast generated by the residual model is poor and use a simple default model (e.g. an integrator). The quality of the forecast can be monitored using the mean-squared error (MSE) in the forecasts. For example, the forecast is poor if the MSE exceeds a user specified value (expected noise variance on the controlled process output).
3) Select the parameters (ι, κ and tr) in ILS.
4) Change the order of the residual model. Note that the residual model is a single series model and efficient techniques (cf. lattice methods) are available for its identification, that are recursive in time as well as model order. Therefore it easy to increase/decrease the order of the model on-line.

CUSUM test

Load-disturbances result in a shift in the mean of the residuals and can be detected reasonably fast using a CUSUM test. The CUSUM test is a statistical method that is widely used in quality control applications to detect small changes in the distribution (e.g. mean) of a quality characteristic and maintain a tight control over a process. It is discussed, e.g., in Gilchrist (1977) and Woodall (1986).

The two-sided CUSUM procedure may be performed as follows (Woodall (1986)): Samples from $\{\hat{x}'(t)\}$ of size n are taken successively and sample means $\{\overline{x}_i, i\leq t\}$ are calculated. It is assumed that the sample means are mutually independent and are normally distributed, i.e. $\overline{x}_i \cong N(\mu_i,\sigma^2)$, where σ^2 is assumed to be known and to remain constant. Assume that $\mu_i=\mu$, $i\leq t$ and the condition $\mu=\mu_0$ is to be maintained. This can be be interpreted in the present context as the detection of the situation where the mean-level of $\{\hat{x}'(t)\}$ has shifted; μ_0 is an estimate of the current mean-level of the residual sequence and since this is unknown *a priori*, a reasonable estimate of this value for use in the test can be obtained by using the moving average of all the sample means, i.e $\mu_0 \cong \frac{1}{n}\sum_{i=t-n+1}^{t} \bar{x}_i$. Then define the cumulative sums (or CUSUMs) as:

$$S_i = \max(0,\ S_{i-1}+\bar{Z}_i-K)$$

$$T_i = \min(0,\ T_{i-1}+\bar{Z}_i+K),\ i=1,\ 2,\ \ldots$$

where $\bar{Z}_i := (\bar{x}_{t-i+1}-\mu_0)/\sigma$, $K \geq 0$, $S_0 := w$, $T_0 := -w$ and $(0 \leq w \leq h)$. The two-sided CUSUM procedure signals an "out-of-control" situation (i.e. the mean-level has shifted) at the first stage N where $S_N \geq h$ or $T_N \leq -h$. If the current time is t, then this means that the shift in the mean-level has occurred at (t-N+1). Usually w=0 but it is recommended to use a head-start value w=h/2 for quick detection of initial "out-of-control" conditions. The CUSUMs (e.g. S_1, S_2, ...) may be calculated at each time t or one can use the CUSUMs calculated at time (t-1) and only compute the latest CUSUM recursively. The latter requires a slight change of notation to indicate the moving time index. The reference value K is usually chosen to be $\theta_1/2$ where θ_1 is the smallest shift that is considered important enough to be detected quickly. The values n, K and h must be selected to suit the needs of a particular application and can be done using a few simulations. Since the procedure also indicates the time (t-N+1) where the disturbance occurred, the process model parameters could be reset to their values before the disturbance has had any effect on them, i.e. $\hat{\theta}(t)$ is set to $\hat{\theta}(t-N)$. Note that the estimation of the process model goes on for N intervals after the disturbance has occurred.

Two major criticisms of the CUSUM test must be noted regarding its sensitivity in tracking the shift in the underlying mean of a process: 1) CUSUM may be too sensitive to small shifts in the mean; 2) it does not detect large shifts in the mean as quickly as the Shewhart-chart (see Gilchrist (1977)). One can handle the second criticism by combining a Shewhart-chart with the CUSUM procedure. The combined procedure produces an out-of-control signal if a single values $\bar{Z}_i$ exceeds a specified control limit, even if the CUSUM procedure does not produce a signal.

Note that the rules stated above for controlling the two-stage identification are entirely heuristic and are based on stability and performance considerations. The CUSUM procedure was tested in simulations to give excellent results. Since the rules are so simple, they were implemented using FORTRAN code. However, in a practical commercial control system the rule base would be much larger and an expert system would be desirable to facilitate development, testing and maintenance of the knowledge base.

6.8 Convergence properties of MAPC

The stability and performance of MAPC depend in a complicated way on the convergence of the Kalman filter predictor and the stability of the predictive controller plus the parameter estimates. If the Kalman filter predictor converges and gives asymptotically offset-free predictions of the output then the predictive controller ensures that there is no steady state control offset. The dynamic performance of the controller depends on its parameters (e.g. N_1, N_2, N_u and the weighting matrices Γ_y and Γ_u). The dynamic behaviour of the predictor depends on the initial conditions (i.e. P(0/-1)) and the noise covariances. No concrete results regarding the stability of the composite system are available at this stage. However, some remarks that are helpful in understanding the overall stability may be made as follows.

The convergence properties of the Kalman filter are discussed in Goodwin and Sin (1984) and Kumar and Varaiya (1986). Essentially the asymptotic behaviour (i.e. convergence and boundedness of the state prediction error covariance matrix, P(t+1/t), and its dependence on its initial value, P(0/-1)) of the Kalman filter predictor when the state-space model is time-invariant is important. The boundedness depends on the observability of the pair (Φ,H). The needed result is summarized in the following theorem.

Theorem (due to Kumar and Varaiya (1986))

Let Q be a square-root of R_v (i.e. $QQ^T=R_v$).

Suppose (Φ,Q) is reachable, (Φ,H) is observable and $R_v>0$.

Then $\lim_{t\to\infty} P(t+1/t)=\overline{P}$ and $\overline{P}$ is the unique, non-negative definite solution of the steady state or algebraic Riccati equation:

$$\overline{P}=\Phi\{\overline{P}-\overline{P}H^T(H\overline{P}H^T+R_v)^{-1}H\overline{P}\}\Phi^T+R_v$$

In particular $\overline{P}$ is independent of the initial covariance P(0/-1).

Thus if the conditions of the theorem hold, which is not hard to verify, then the Kalman filter predictor converges to a time-invariant predictor with the steady state covariance given by $\overline{P}$ and the steady state gain $\overline{K}$ given by equation (35).

In the time-varying or self-tuning case it is obvious that in order to give proper estimates of the states (or predictions of the output), the estimates of the parameters of the process and disturbance models should converge to those of the actual process subjected to disturbances in some sense (e.g. in frequency domain terms). In the self-tuning case the Kalman filter predictor gain matrix is computed on the basis of the latest parameter estimates and the covariance P(·) available from the previous time step. If the paramter estimates converge, the Kalman gain converges to some steady state value as $t \to \infty$.

Since the adaptive controller is explicit at least a local stability result can be roughly stated as follows: If the Kalman filter predictor converges and the basic controller is stable with initial estimates of the parametes of the process and disturbance models, i.e. $\hat{\theta}(0)$ and $\hat{\theta}_r(0)$, and the converged (or steady state) predictor, then it is reasonable to expect that the time-varying adaptive controller would be stable provided that $\hat{\theta}(t)$ lies in a stable region around $\hat{\theta}(0)$. Note that $\hat{\theta}_r(t)$ influences only the forecast of the disturbances and hence the disturbance rejection, and does not affect the stability. The stability of the controller depends only on the process model, i.e.

$\hat{\theta}(t)$. The parameter projection feature of ILS can be used to ensure that $\hat{\theta}(t)$ stays in a stable region around $\hat{\theta}(0)$ or some other nominal model for which the scheme is found stable, for all time. For simple processes (e.g. 1st or 2nd order) it is relatively easy to define such regions.

6.9 Literature review

A number of self-tuning controllers are in existence and a few of the most popular versions are briefly reviewed below to emphasize the advantages of MAPC.

STR

Interest in self-tuning control really started with the introduction of the self-tuning regulator (STR) by Astrom and Wittenmark (1973). The STR asymptotically provides minimum variance performance, and although it is a very useful technique for a wide variety of processes, it suffers from the following drawbacks: it is sensitive to a wrong assumption of the process dead-time; it results in excessive control action (i.e. no control weighting); it has no performance flexibility or tuning knobs; it does not have setpoint tracking ability; and it cannot handle general load-disturbances. It is also not directly applicable to NMP processes as the control-law would then have unstable poles which would make the closed-loop unstable. Note that the minimum variance control-law cancels the process poles and zeros.

STC

The self-tuning controller (STC) proposed by Clarke and Gawthrop (1979) improves the STR by the introduction of a more general controller cost function. This leads to setpoint tracking ability, performance flexibility (i.e. control action can be penalized) as well as the ability to stabilize NMP processes through the use of P, Q and R weighting transfer functions in the cost function. But this method also suffers from sensitivity to a wrong assumption about the process dead-time. However, the way it is normally formulated, it is not very easy to tune on-line and also it does not handle general load-disturbances, although it gives asymptotically offset free regulation if increments of u are penalized by an appropriate choice of Q. Note that in STC over- or under-parameterization of the process model is usually not a problem. Because of the more general cost function used by STC, it is also referred to as a generalized minimum variance (GMV) controller. Various interpretations of the GMV controller can be found in Gawthrop (1977).

GPP

Another popular approach is the pole placement controller (e.g. see Zanker and Wellstead (1979), Astrom and Wittenmark (1980) and Puthenpura and MacGregor (1986)) which allows the poles of the closed-loop system to be placed at arbitrary locations. The zeros of the process may or may not be cancelled. In this approach the process model is usually identified explicitly. The main advantage of the pole placement approach is the simplicity with which desired closed-loop servo performance can be obtained. All that is needed is the specification of the closed-loop characteristic polynomial. The pole placement method overcomes the problem of a wrong specification of the process dead-time (by over-parameterizing the B-polynomial), but needs an exact knowledge of the process model order. If A and B are both over-parameterized, then this leads to problems in identification. The estimated A and B polynomials may then contain pole-zero cancellations which cause problems in control. Recently a generalized poleplacement algorithm was proposed by Lelic *et al.* (1987) which belongs to a class of multi-step predictive controllers based on optimization approach. Here closed-loop pole placement is achieved via a multi-step cost function. Its advantages are improved control performance and easier tuning. Note that the pole placement method is not well suited for regulatory control applications.

GPC

The GPC proposed by Clarke *et al.* (1987a, 1987b) is based on a long-range prediction of the process output and can be considered as a generalization of the GMV control-law. It is robust in the face of an incorrect specification of process dead-time (due to the multi-step cost function it uses) and can work even when the process model is over- or under-parameterized. It has simple tuning knobs that can be used to obtain desired control performance or to stabilize processes with complex dynamics (cf. unstable zeros). However, it is based on a CARIMA model of the process subjected to disturbances, which amplifies the effect of noise in the measured signals in both estimation and control. Also it approximates load-disturbances using a random-walk model and does not model load-disturbances that have a more general structure.

LQG

The adaptive LQG is quite popular and it also like GPC is based on a multi-stage cost function. It is a very robust controller with well proven theoretical stability results. It can handle incorrectly specified process dead-time but is sensitive to a wrong parameterization of the process model just like the pole placement method. A very general LQG (i.e. based on a general model) was developed by M'Saad *et al.* (1986). Clarke *et al.* (1987c, 1987d) have developed an LQG controller based on a CARIMA model of the process.

IDCOM and DMC

The ideas of using long-range predictive control and the ability to follow operator specified reference trajectories first originated in the industrially successful non-adaptive control schemes like IDCOM (Richalet *et al.* (1978)) and DMC (Cutler and Ramaker (1980)). The idea of receding-horizon control is also due to these methods. Their success in the industry has lead to the incorporation of these ideas into adaptive control. One example is GPC. The MAPC presented in this chapter is motivated by the desire to develop an adaptive version of MOCCA which is similar to IDCOM or DMC and which is presented in Chapter 3.

6.10 Simulated, single-input, single-output, applications

Simulated examples are discussed in this section which illustrate selected features of the proposed scheme. The discussion of the examples is organized in terms of features. Where appropriate, comparisons with GPC are included. The details of the examples are presented in Appendix 1 to permit easier comparison and reference.

Handling noise: Process measurements are typically noisy. If the measurement noise is not taken into account in any practical control scheme the control signal may exhibit excessive input/output variation which is not desirable. The Kalman filter prediction used in MAPC optimally rejects measurement noise and passes the minimum variance estimates of the output to the controller. Example 1 shows the performance of the basic proposed controller (i.e. non-adaptive) and that of basic GPC. The GPC is based on a CARIMA model of the process subjected to disturbances and because of the incremental formulation, it is very sensitive to noise in the measured signals. However, GPC models structured noise through the C-polynomial for minimum variance prediction and hence by an appropriate choice of C it can be made to reject the measurement noise. But this is more difficult to do in GPC than in MAPC as the Kalman filter predictor provides a direct characterization of the measurement noise through the noise covariances. The results presented in the example show that the proposed scheme performs significantly better (i.e. both u and y variations are considerably less). As pointed out, the performance of GPC can be improved by using a noise observer polynomial. The Kalman filter prediction based MAPC on the other hand gives significantly improved performance with almost no effort.

The adaptive case is illustrated by example 3. The performance is once again similar to the non-adaptive case. Note however that the parameter estimates are smoother in MAPC, presumably because u and y show less variation compared with GPC.

Handling load-disturbances: If an exact internal model of the load-disturbances is used, then the proposed controller can reject these disturbances in a dead-beat manner. GPC models disturbances as random steps at random times (i.e. uses a simple random-walk model) and hence if the disturbances have structure (i.e. vary more slowly than can be modelled adequately by a sequence of steps) can show poor performance. This is illustrated in example 2 for the non-adaptive case. A comparison of Figures 6.12 and 6.13 shows that the use of a complete disturbance internal model is much better than the use of a simple integrator in MAPC. For example use of the exact internal model of the load-disturbance could result in a dead-beat cancellation of the disturbance (k+1) time steps after its occurrence where k is the process pure dead-time. Note that the magnitudes of the spikes in y(t) using MAPC is determined by the form of the disturbance and the dead-time.

The adaptive case is a bit more complicated. When a disturbance occurs, if the process model identification is not stopped then assuming that the disturbances have a different frequency content than the process dynamics, the process model would be corrupted and the subsequent control performance may deteriorate or may be unstable. Hence detecting the occurrence of a load-disturbance and switching off the process model identification and/or resetting the process model parameters to the previous best set is very important. A CUSUM test on the residual sequence is used to detect the occurrence of a load-disturbance in MAPC. Example 4 illustrates the performance of MAPC in the presence of load-disturbances. When the residual model is poor initially, the rejection of load-disturbances is also correspondingly poor. However, as the identification of the residual model improves the disturbance rejection also improves.

Setpoint tracking: The setpoint tracking ability of the proposed controller depends on the process model as well as the residual forecast and the controller parameters. All the examples demonstrate the excellent servo performance of the proposed scheme. For example, dead-beat tracking is obtained when $N_y=N_u$, $\Gamma_y=I$ and $\Gamma_u=0$. As the output horizon increases the response becomes slower. Ability to use a trajectory for the reference input is a special feature of the controller. This is possible due to the multi-step cost function.

Handling unmodelled dynamics: The proposed method offers a way to treat or include unmodelled effects explicitly into the prediction via the residual model. If the process model gives stable control then the unmodelled dynamics are picked up by the residual model and included in the prediction of $\hat{y}(\cdot)$. By using a sufficiently high-order residual model dynamics of the process can be completely modelled and compensated. Example 5 illustrates the performance of the proposed scheme when there are significant unmodelled dynamics due to the use of a reduced order process model. GPC failed to give stable control in this case for the same initial conditions and the controller cost function parameters, and this indicates a lack of robustness.

In summary, the simulation examples presented here show that MAPC performs as expected and can give excellent results. Further simulation study is required to fully evaluate MAPC vis a vis GPC and other related schemes.

6.11 Conclusions

The proposed Multi-step Adaptive Predictive Controller (MAPC) is based on the integration of the following key ideas: 1) use of a long-range prediction over a finite output horizon, normally beyond the process dead-time; 2) choice of a control horizon beyond which the control action increments are assumed to be zero; 3) consideration of weighting on tracking error and control action; 4) ability to follow operator specified reference trajectories; 5) use of a state-space model in controller synthesis; 6) use of multiple models in estimation; and 7) use of a two-stage identification scheme.

The use of an output horizon makes the controller less sensitive to the specification of the process dead-time and allows processes with complex dynamics (cf. NMP zeros, unstable or complex poles) to be controlled satisfactorily. The output horizon, the control horizon and the weighting matrices provide simple tuning parameters that can stabilize any process and achieve any user specified control performance. In most cases default settings can be used for these parameters.

The use of a state-space model in controller synthesis is a key feature of the proposed scheme. The state-space model is used to divide the total process response into two parts: an input/output response (or process dynamics) part and a residual part. The residual model represents the unmodelled residuals in the process response such as structured noise, load-disturbances and unmodelled dynamics which can be predicted and eliminated using the predictive controller to give asymptotically offset-free regulation. The advantage is that these models can be estimated separately and substituted separately into the state-space model to give a complete characterization of the process response. In addition the state-space model allows explicit modeling of measurement noise which is optimally rejected using a Kalman filter.

In most self-tuning control schemes a disturbance model is estimated on-line along with process model parameters. Successful identification of a disturbance model in such a single-stage identification method is usually difficult because the disturbance characteristics vary with time and most estimation methods suffer from an inherent weakness of slow convergence of the estimated disturbance model.

The proposed method is an alternative where a process model and a residual model are estimated separately and provide a great deal of flexibility. The residual model is usually a fast or higher order model and since it is identified separately from the process model it overcomes the problem of slow convergence associated with a single-stage identification scheme. Also there is no need to use fixed, *a priori* chosen polynomials in the disturbance model.

A simple rule base was derived to identify conditions under which each model can be estimated and was tested in a conventional programming framework. In a real application these rules are best imlemented as an expert system.

A set of preliminary SISO simulation examples indicate excellent performance and justify further evaluations and extension to the multivariable case.

References

Arzen, K. E. (1986): "Use of Expert Systems in Closed Loop Feedback Control", Proc. American Control Conference, Seattle, Wa, USA, 140.

Astrom, K. J. (1970): Introduction to Stochastic Control Theory, Academic Press.

Astrom, K. J. (1983a): "Theory and Applications of Adaptive Control - A Survey", Automatica, 19, 5, 471.

Astrom, K. J. (1983b): "Analysis of Rohrs' Counter Examples to Adaptive Control", Proc. 22nd Conf. on Decision and Control, San Antonio, Tx, USA, 982.

Astrom, K. J. (1986): "Adaptation, Auto-tuning and Smart Controls", Proc. CPC-III.

Astrom, K. J. and B. Wittenmark (1973): "On Self-tuning Regulator", Automatica, 9, 185.

Clarke, D. W. (1967): "Generalized Least Squares Estimates of the Parameters of a Dynamic Model", Paper 3.17, Preprints IFAC Symposium on Identification, Prague.

Clarke, D. W. and P. J. Gawthrop (1979): "Self-tuning Control", Proc. IEE, 126, 633.

Clarke, D. W., C. Mohtadi and P. S. Tuffs (1987a): "Generalized Predictive Control; Part 1: The Basic Algorithm", Automatica, 23, 2, 137.

Clarke, D. W., C. Mohtadi and P. S. Tuffs (1987b): "Generalized Predictive Control; Part 2: Extensions and Interpretations", Automatica, 23, 2, 149.

Clarke, D. W., P. J. Kanjilal and C. Mohtadi (1984c): "A Generalized LQG Approach to Self-tuning Control; Part 1: Aspects of Design", O. U. E. L. Report.

Clarke, D. W., P. J. Kanjilal and C. Mohtadi (1984d): "A Generalized LQG Approach to Self-tuning Control; Part 2: Implementation and Simulation", O. U. E. L. Report.

Cutler, C. R. and B. L. Ramaker (1980): "Dynamic Matrix Control - A Computer Control Algorithm", Proc. JACC, San Fransisco, USA, WP5-B.

Freidland, B. (1978): "Notes on Separate Bias Estimation", IEEE Trans. Aut. Contr., AC-23, 4, 795.

Garcia, C. E. and M. Morari (1982): "Internal Model Control.

1: A Unifying Review and Some New Results", I&EC Process Des. Dev., 21, 2, 308.

Gawthrop, P. J. (1977): "Some Interpretations of the Self-tuning Controller", Proc. IEE, 124, D, 889.

Gertler, J. and H. Chang (1986): "An Instability Indicator for Expert Control", IEEE Control Systems Magazine

Gilchrist, W. (1977): Statistical Forecasting, Wiley-Interscience.

Godfrey, L. and K. P. Jones (1986): Signal Processing in Control, Springer-Verlag.

Goodwin, G. C., D. J. Hill and M. Palaniswami (1984): "A Perspective on Convergence of Adaptive Control Algorithms", Automatica, 20, 5, 519.

Goodwin, G. C. and K. S. Sin (1984): Adaptive Filtering, Prediction and Control, Prentice-Hall.

Goodwin, G. C. (1986): "Digital Adaptive Control (A Robust Approach)", CPC-III.

Iserman, R. and K. H. Lachmann (1985): "Parameter-adaptive Control with Configuration Aids and Supervision Functions", Automatica, 21, 6, 625.

Lam, K. P. (1982): "Design of Stochastic Discrete Time Linear Optimal Regulators", Int. J. Systems Sci., 13, 9, 979.

Lam, K. P. (1985): "On a Self-tuning Controller with Retained and Changeable Memory", Proc. 24th CDC, 1221.

Lelic, M. A. and M. B. Zarrop (1987): "Generalized Pole Placement Self-tuning Controller. Part 1: Basic Algorithm", Intl. J. Control, 46, 2, 457.

MacGregor, J. F., T. J. Harris and J. D. Wright (1984): "Duality Between the Control of Processes Subject to Randomly Occuring Deterministic Disturbances and ARIMA Stochastic Disturbances", Technometrics, 26, 4, 389.

M'Saad, M., M. Duque and I. D. Landau (1986): "Practical Implications of Recent Results in Robustness of Adaptive Control Schemes", Proc. 25th Conf. on Decision and Control, Athens, Greece, 477.

M'Saad, M., M. Duque and I. D. Landau (1986): "An LQG Adaptive Controller for Industrial Processes", Proc. ACC, Seattle, Wa, USA, 1097.

Ortega, R., M. M'Saad and C. Canudas (1986): "Practical Requirements and Theoretical Results in Robust Adaptive Control", Proc. ACC, Seattle, Wa, USA, 86.

Puthenpura, S. C. and J. F. MacGregor (1986): "Pole-zero Placement Controllers and Self-tuning Regulators with Better Setpoint Tracking", Report #1005, Depart. of Chemical Engrg., McMaster University, Hamilton, Canada.

Richalet, J. A., A. Rault, J. L. Testud and J. Papon (1978): "Model Predictive Heuristic Control: Applications to Industrial Processes", Automatica, 14, 413.

Seborg, D. E., T. F. Edgar and S. L. Shah (1986): "Adaptive Control Strategies for Process Control: A Survey", AIChE J., 32, 6, 881.

Schumann, R., K. H. Lachmann and R. Isermann (1982): "Towards Applicability of Parameter-Adaptive Control Algorithms", Proc. 8th IFAC World Congress, Kyoto, Japan.

Shah, S. L. and W. R. Cluett (1987): "RLS Based Estimation Schemes for Self-tuning Control", Proc. Annual AIChE Meeting, New York, NY.

Sinha, N. K. and B. Kuszta (1983): Modeling and Identification of Dynamic Systems, Van Nostrandt.

Sanoff, S. P. and P. E. Wellstead (1986): "Expert Identification and Control", Proc. IFAC Conf. on Identification and System Parameter Estimation, York, UK.

Smith, O. J. M. (1957): "Closer Control of Loops with Dead Time", Chem. Engg. Progr., 53, 217.

Walgama, K. S. (1986): "Multivariable Adaptive Predictive Control for Stochastic Systems with Time Delays", M. Sc. Thesis, Depart. of Chemical Eng., University of Alberta.

Wellstead, P. E., D. Prager and P. Zanker (1979): "A Pole-assignment Self-tuning Regulator", Proc. IEE, 126, 781.

Wittenmark, B. and K. J. Astrom (1984): "Practical Issues in the Implementation of Self-tuning Control", Automatica, 20, 5, 595.

Woodall, W. H. (1986): "The Design of CUSUM Quality Control Charts", Journal of Quality Technology, 18, 2, 99.

Ydstie, B. E. (1984): "Extended Horizon Adaptive Control", Proc. IFAC 9th World Congress, Budapest, Paper 14.4/E-4.

Appendix 6.1

SISO Simulation examples

The following two process models were used in the simulations.

System (1): The input/output relation of the process is given by,

$$y(t) - 0.7y(t-1) = u(t-4)$$

The load-disturbances are assumed to be random exponentials given by,

$$d(t)=\frac{1}{1-1.95q^{-1}+0.95q^{-2}}e(t)=\frac{1}{(1-q^{-1})(1-0.95q^{-1})}e(t)$$

where {e(t)} is non-zero only at isolated instants of time. The state-space model for this system has the form:

$$(1)\ z(t+1) = \begin{bmatrix} \Phi_1 & 0 \\ 0 & \Phi_2 \end{bmatrix} z(t) + \begin{bmatrix} \Lambda_1 \\ 0 \end{bmatrix} u(t) + w(t)$$

$$y(t) = Hz(t) + v(t)$$

where,

$$\Phi_1 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & .7 \end{bmatrix}$$

$$\Phi_2 = \begin{bmatrix} 0 & 1.95 \\ 1 & -.95 \end{bmatrix}$$

$$\Lambda_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\Lambda_2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$H = [0\ 0\ 0\ 0\ 1\ 0\ 1]$

$w = [w_1\ 0\ 0\ 0\ 0\ e]^T$

$\{w_1(t)\}$ and $\{v(t)\}$ are zero-mean, white noise sequences; $\{e(t)\}$ is zero except at isolated instants of time.

System (2): $A(q^{-1})\ y(t) = B(q^{-1})\ u(t-1)$

Where,

$A(q^{-1}) = 1-2.39q^{-1}+2.691q^{-2}-1.966q^{-3}+0.6703q^{-4}$

$B(q^{-1}) = 0.6785+6.322q^{-2}+5.831q^{-3}+0.5334q^{-4}$

Summary of Examples:

(i) Handling noise:
  non-adaptive case (example 1)
  adaptive case (example 3)

(ii) Handling disturbances:
  non-adaptive case (example 2)
  adaptive case (example 4)

(iii) Model-process mismatch:
  adaptive case (example 5)

EXample 1

Objective: To show how the basic, non-adaptive MAPC performs in the presence of process and measurement noise and to compare it with the basic GPC.

System: (1)

System forcing or disturbances: $\{w_1(t)\}$ and $\{v(t)\}$ have a standard deviation of 0.1; $\{w_2(t)\}$ is 0 $\forall t$ (i.e. no load-disturbances).

KF parameters: $P(0/-1)=0.1I$; $R_v=0.1I$ and $R_v=0.1$

Controller parameters: $N_1=4$, $N_2=5$, $N_u=1$, $\Gamma_y=I$ and $\Gamma_u=0$ (for MAPC);

$N_1=4$, $N_2=5$, $N_u=1$ and $\lambda=0$ (for GPC).

Results/observations:

a. Both u and y are significantly better in the case of the proposed scheme than with GPC (see Figures 6.10 and 6.11).

b. This example shows the advantage of using Kalman filter prediction in the proposed scheme. Since the Kalman filter prediction optimally rejects noise in the signals going to the predictive controller, the controller is not sensitive to measurement noise. The incremental model employed in GPC amplifies the effect of noise on the signals used by the controller and as a result gives very poor results compared with MAPC. Note that by using an appropriate C polynomial, GPC can be made equivalent to MAPC, but it is not immediately clear how the user could choose the correct C. In the present case, C is taken to be 1. MAPC on the other hand gives a satisfactory performance with almost no effort (i.e. requires only the specification of the noise covariances).

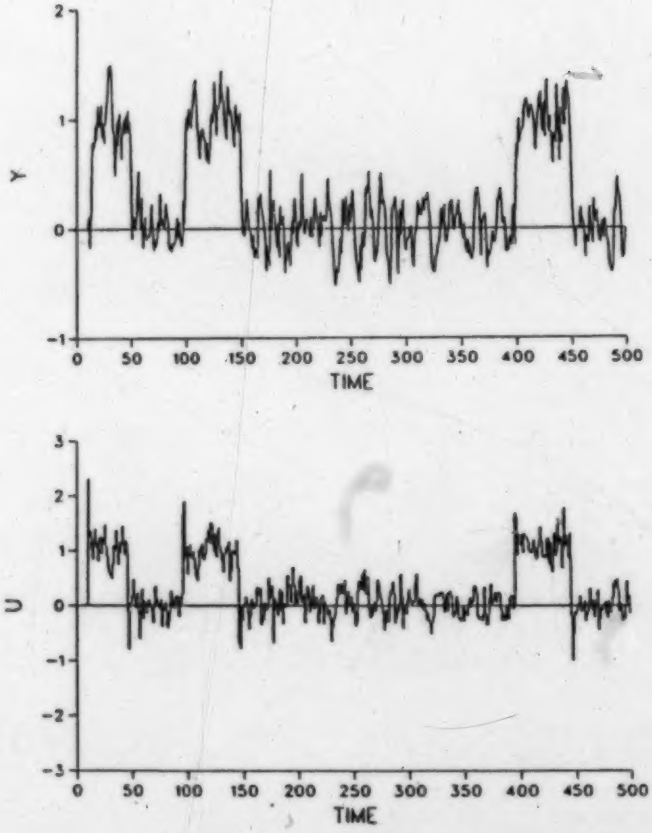


Figure 6.10 Performance of the basic, non-adaptive MAPC in the presence of noise

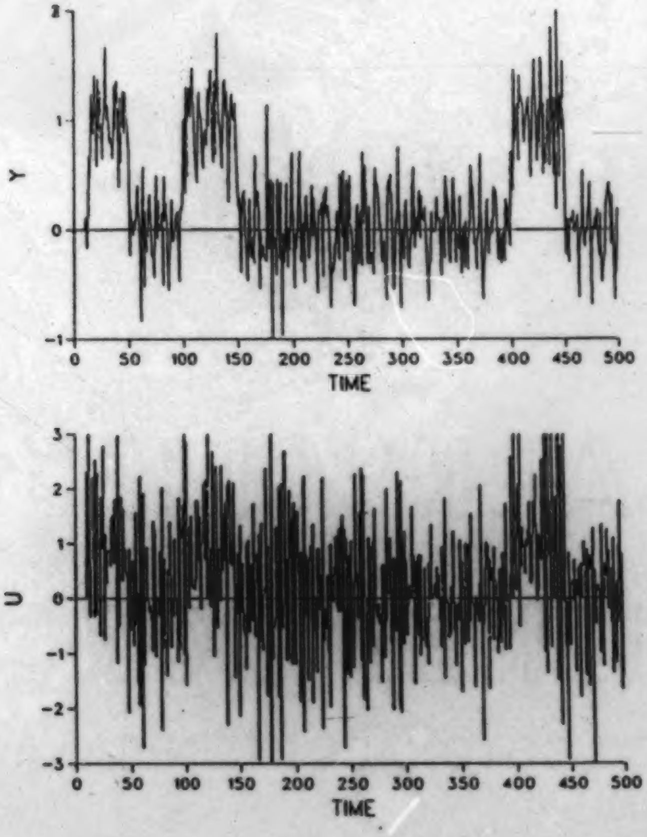


Figure 6.11 Performance of the basic, non-adaptive GPC in the presence of noise

Example 2

Objective: To show the advantage of using an internal model of the load-disturbances in MAPC.

System: (1)

System forcing or disturbances: $\{w_1(t)=0\}$ and $\{v(t)=0\}$ (i.e. no noise)

$w_2 \neq 0$ only at isolated instants of time (i.e. disturbances occur randomly but well isolated); the disturbances are assumed to be random exponentials.

KF parameters: $P(0/-1)=R_w=0.1I$ and $R_v=0.1$

Controller parameters: $N_1=4$, $N_2=5$, $N_u=1$, $\Gamma_y=I$ and $\Gamma_u=0$ (for MAPC);

$N_1=4$, $N_2=5$, $N_u=1$ and $\lambda=0$ (for GPC).

Results/observations:

a. If a correct internal model of load-disturbances is used then the load-disturbances can be exactly cancelled in a dead-beat manner one step after the process dead time, if the controller is dead-beat. In the present case, since $N_y=2$, the controller is not dead-beat, but almost close to it. The resulting performance is shown in Figure 6.12. If a simple integrator is used, the steady state offset is eliminated but the dynamic performance is poor (see Figure 6.13).

b. GPC uses a *simple integrator* in the control-law so that it is able to eliminate the steady state offset (Figure 6.14) in a manner similar to that of MAPC using a simple integrator model of the disturbances (cf. Figure 6.13). However, the dynamic performance of GPC is not as satisfactory as when an exact internal model is employed (cf. Figures 6.12 and 6.14). In section 6.10 it was pointed out that a disturbance model can be incorporated into GPC by a proper specification of the C polynomial. However, in general this is more difficult to do than with MAPC.

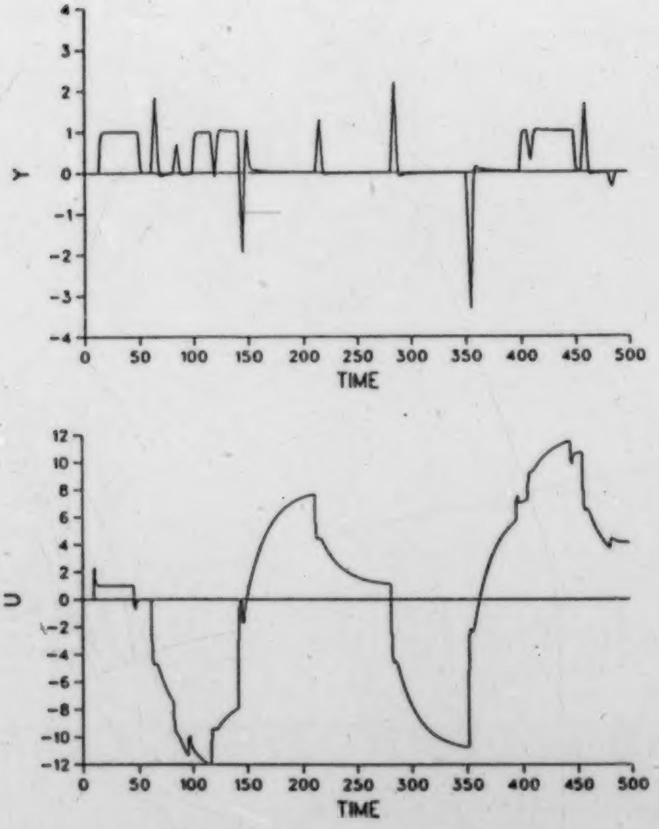


Figure 6.12 Performance of the basic, non-adaptive MAPC in the presence of load-disturbances (an exact internal model is used)

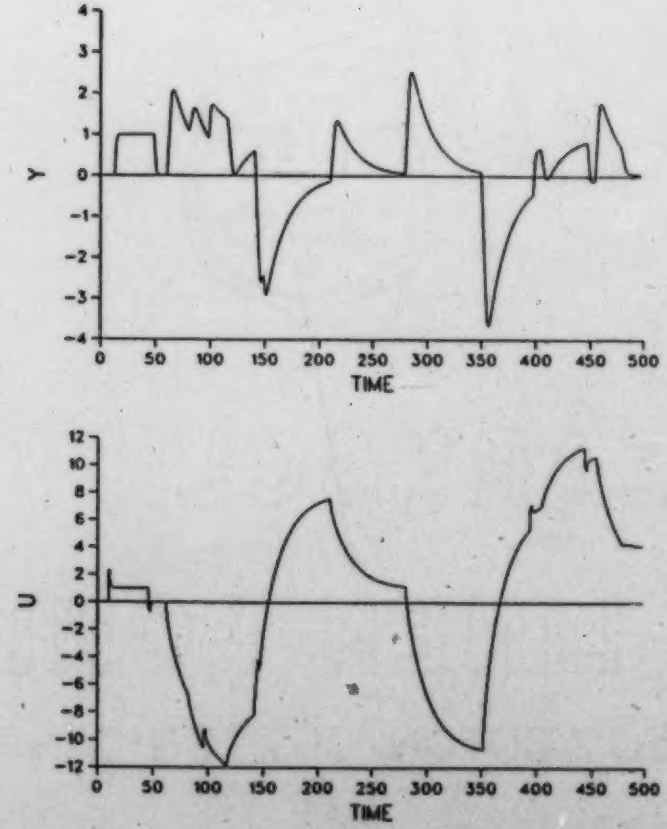


Figure 6.13 Performance of the basic, non-adaptive HAPC in the presence of load-disturbances (the disturbance model is an integrator)

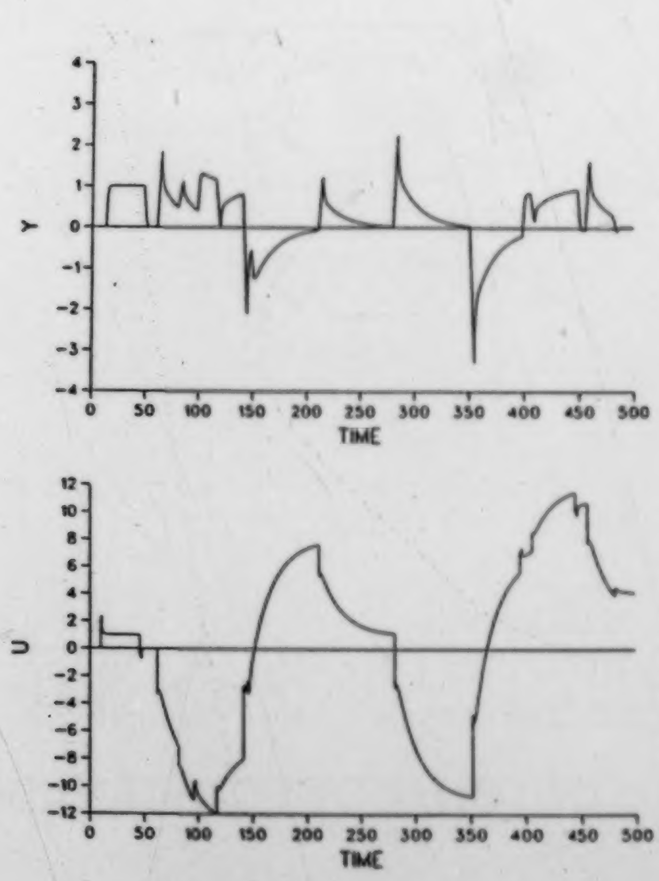


Figure 6.14 Performance of the basic, non-adaptive GPC in the presence of load-disturbances

Example 3

Objective: To show the performance of the adaptive version of MAPC and to compare it with that of adaptive GPC in the presence of noise.

System: (1)

System forcing or disturbances: $\{w(t)=0\}$, i.e. no process noise or sustained disturbances; $\{v(t)\}$ is a white noise sequence of standard deviation 0.1 (i.e. only measurement noise is present).

KF parameters: $P(0/-1)=R_v=0.1I$; $R_v=0.1$.

Controller parameters: $N_1=4$, $N_2=5$, $N_u=1$, $\Gamma_y=I$ and $\Gamma_u=0$ (for MAPC);

$N_1=4$, $N_2=5$, $N_u=1$ and $\lambda=0$ (for GPC).

Parameter estimation: ILS (with extended least squares option).

$(1+a_1q^{-1})y(t) = b_1u(t-4)+(1+c_1q^{-1})e(t)+d'(t)$

a_1, b_1 and c_1 are estimated.

$\kappa=10.0$, $\iota=0.1$, tr=3 $(P(0)=S(0)=P(0)^{-1}=I$.

$\hat{\theta}(0)=(-.5,\ .5,\ 0.)$ (= $(.7413,\ 1.67,\ 0.)$ when normalized).

$\theta_r(0)=(1,1)$; the residual model is not estimated as there are no load-disturbances.

KF gain is calculated at each time with latest estimates of the parameters and old values of the various covariances.

Results/observations:

a. MAPC performs significantly better than GPC (cf.

Figures 6.15 and 6.17). Note that only measurement noise is present. The performance of GPC can perhaps be improved by using C=A, but in the present case C is taken to be 1.

b. The parameter estimates behaved better in the case of MAPC as they depend on the input/output data which was less influenced by noise in MAPC than in GPC.

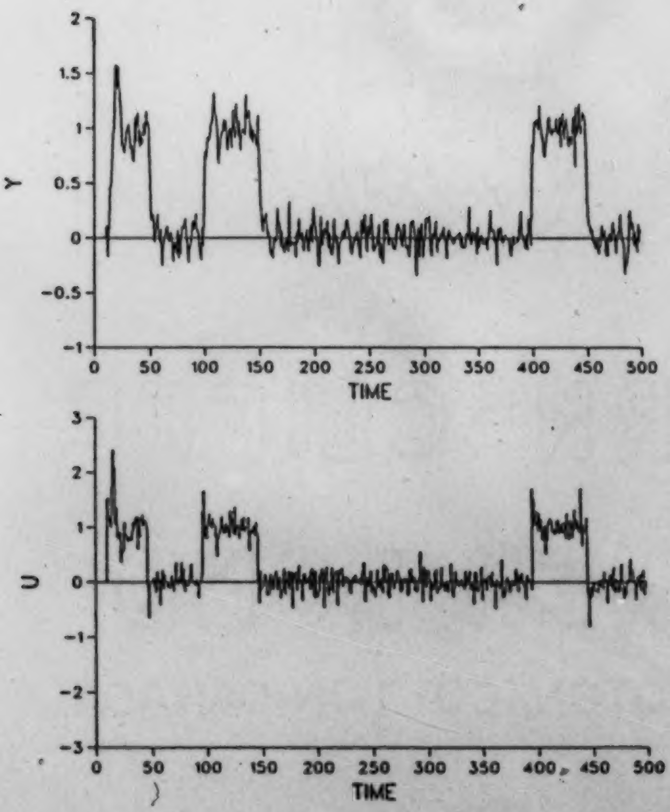


Figure 6.15 Performance of MAPC in the presence of noise

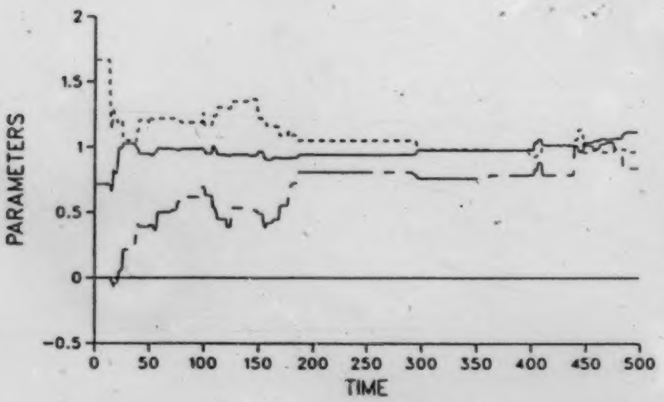


Figure 6.16 Parameter estimates in MAPC (the parameter values are normalized)

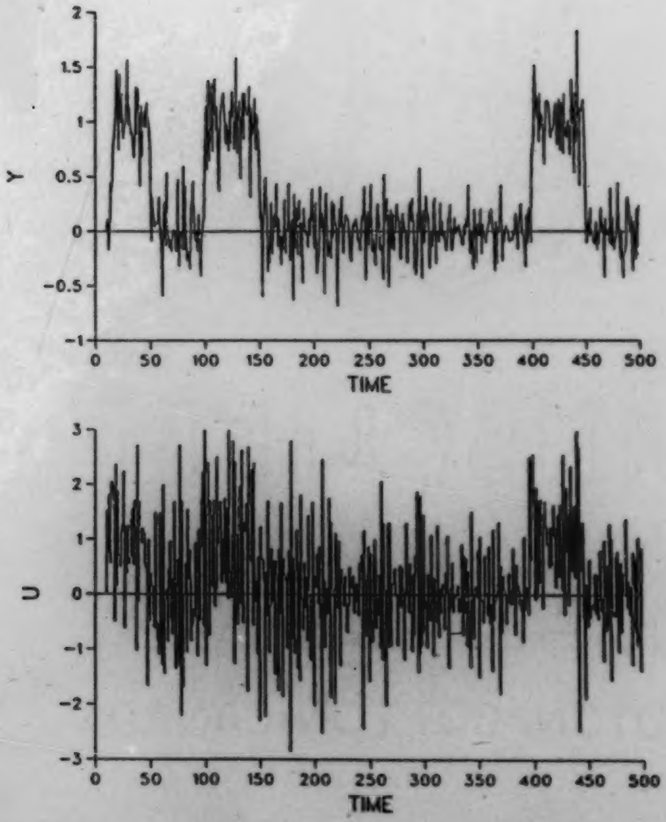


Figure 6.17 Performance of GPC in the presence of noise

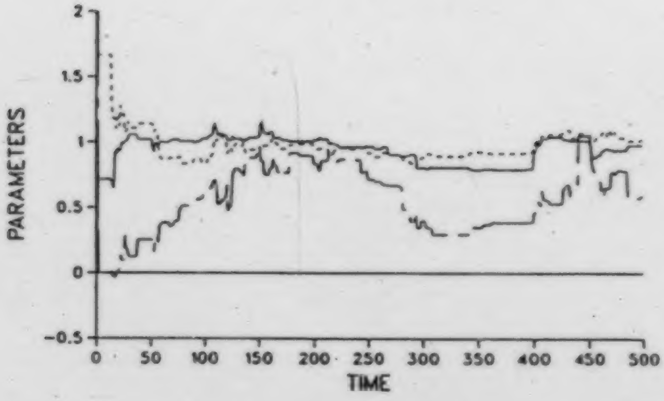


Figure 6.18 Parameter estimates in GPC (the parameter values are normalized)

Example 4

Objective: To illustrate the performance of MAPC in the presence of load-disturbances. Note that a residual model is identified after the identification of a process model. This example also shows the effectiveness of the AI rules used to detect the disturbances and switch off the identification of the process model.

System: (1)

System forcing or disturbances: $\{w_1(t)=0\}$ and $\{v(t)=0\}$, i.e., no noise; $w_2 \neq 0$ only at isolated instants of time; the load-disturbances occur as random exponentials.

KF parameters: $P(0/-1)=R_v(0)=0.1I$, $R_v=0.1$.

Controller parameters:

Case (i): $N_y=8$, $N_1=1$, $N_u=1$, $\Gamma_y=I$ and $\Gamma_u=0$

Case (ii): $N_y=2$, $N_1=1$, $N_u=1$, $\Gamma_y=I$ and $\Gamma_u=0$

Parameter estimation:

process model: (ARMA) with one A and one B polynomial parameters;

ILS; $\kappa=10^6$, $\iota=0$, tr=2 and $\hat{\theta}(0)=(-.55,.45)$;

Residual model: (Single-series) with 3 parameters; ILS; $\kappa=10^6$,

$\iota=0$, tr=3 and $\hat{\theta}_r(0)=(0.,0.,0.)$.

Results/observations:

a. Setpoint tracking was good and did not deteriorate after the introduction of disturbances (see Figures 6.19 and 6.20, where unit step changes were introduced at t=10,40,90,140,390,440).

b. when disturbances occur (at approximately t=160,210,280,350,405,480), a CUSUM test based on the residuals was used to detect their occurrence and to switch off process model identification and to reset process model parameters to the values before the onset of the disturbance.

c. The residual model identification was on all the time.

d. As can be seen from the Figures 6.19 and 6.20 initially (t<400) disturbance rejection was poor (only offset was eliminated at steady state) but as its structure was identified the performance improved (t>450).

e. The disturbances used in this example (i.e. random exponentials) were too severe and changed too fast for their structure to be identified in a few load-changes. More slowly varying disturbances (e.g. higher order than first) would be eliminated better.

f. When the output horizon, N_y is decreased the performance improved (i.e. responses were faster) since the controller became more like a dead-beat design.

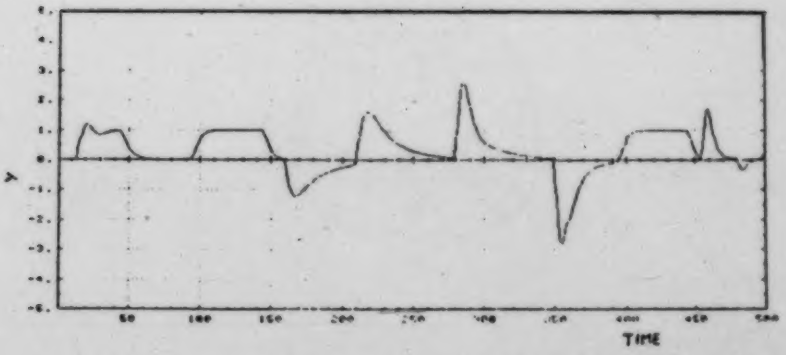


Figure 6.19 Performance of MAPC in the presence of load-disturbances (case (i), N_y=8)

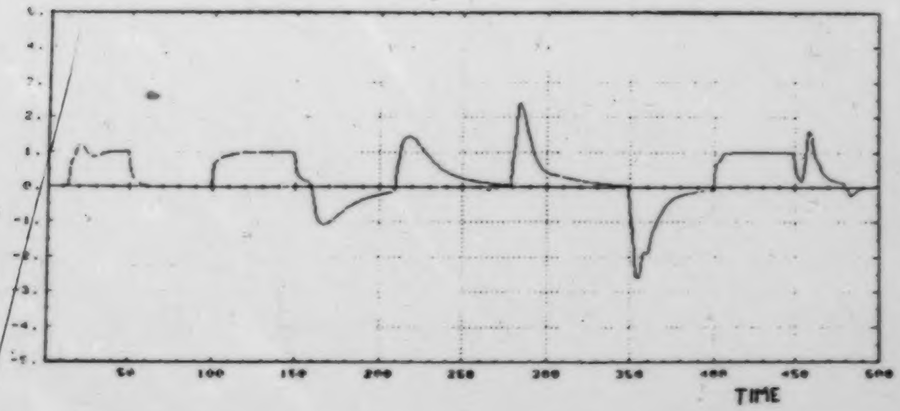


Figure 6.20 Performance of MAPC in the presence of load-disturbances (case (ii), N_y=2)

Example 5

Objective: To illustrate the performance of MAPC in the presence of unmodelled dynamics (no noise or load-disturbances).

System: (2); 4th order.

KF parameters: $P(0/-1)=R_w=0.1I$, $R_v=0.1$.

Controller parameters: $N_y=10$, $N_1=1$, $N_u=1$, $\Gamma_y=I$ and $\Gamma_u=0$.

Parameter estimation:

process model: (2nd order ARMA); ILS; $\kappa=1000$; $\iota=0$; tr=4; input/output data are passed through a low-pass filter $(0.4q^{-1}/(1-0.6q^{-1}))$ to get rid of higher frequencies; $\hat{\theta}(0)=(-1.8,.9,.6,6.)$

Residual model: Single-series with 3 parameters; ILS; $\kappa=1000$; $\iota=0$; tr=15; $\hat{\theta}_{res}(0)=(0.,0.,0.)$

Results/observations:

a. Since MAPC is explicit, the stability is local, i.e. depends on the initial parameters. The non-adaptive case was stable with $\hat{\theta}(0)$ and $\hat{\theta}_r(0)$.

b. The performance was excellent as illustrated. The setpoint changes were tracked excellently (see Figure 6.21) eventhough the MPM was considerable.

c. For the case when the residual model was not estimated (i.e. a simple integrator), the setpoint following was very poor with large variations. The results are not included.

d. The GPC was found to be unstable with the same $\hat{\theta}(0)$ as for MAPC even after a large number of trials with different values of the controller parameters. This shows that the GPC is less robust compared with MAPC. This is perhaps due to the Kalman filter prediction employed by MAPC, which effectively filters out the MPM and provides stable control. This point was discussed in Chapter 6.

e. A higher order residual model (e.g. 4th order) can completely capture the unmodelled dynamics and in combination with the reduced order process model can result in even better setpoint tracking (note that the residual model used was not of sufficiently high order so that its estimates drifted considerably tending to give poorer control as time progressed (cf. t>350)).

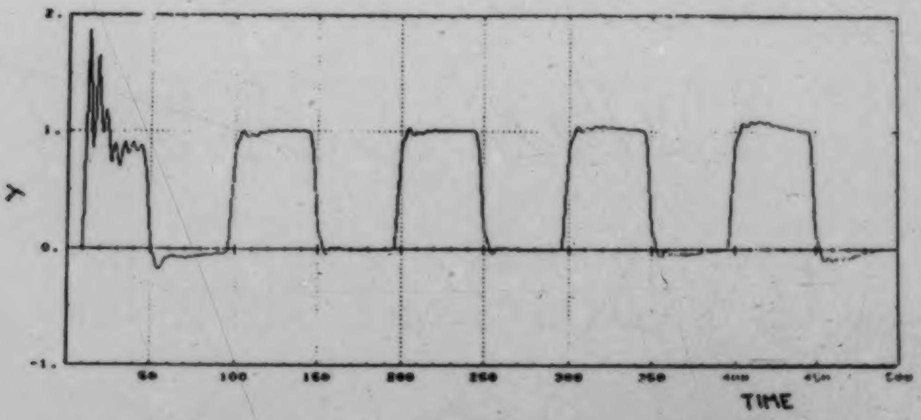


Figure 6.21 Performance of MAPC in the presence of unmodelled dynamics

7. Conclusions and recommendations for future work

7.1 Conclusions

1) The most significant contribution of this thesis is the Multi-step Adaptive Predictive Controller (MAPC). It was developed to solve several practical problems encountered in the application of adaptive control techniques to industrial processes such as time-delays, disturbances, noise and on-line parameter estimation. MAPC can be compared to GPC in structure but is formulated differently and handles noise and disturbances differently. The advantages are improved estimation, prediction and rejection of the disturbances and other unmodelled effects. MAPC can also be thought of as an adaptive version of MOCCA, the non-functional step response model based predictive controller developed in Chapter 3; but MAPC uses a minimal, parametric model so that it can be estimated on-line. In MAPC a process model and a residual model are estimated separately on-line and are incorporated into a state-space model for subsequent use in controller synthesis. The process model represents the dominant dynamics of the process and is required to give a stable and reasonable control of the process. The residual model represents all other unmodelled effects such as noise, load-disturbances and MPM which can be predicted and eliminated using the predictive controller. The two-stage identification used in MAPC

is an alternative to a single-stage identification used in almost all adaptive control algorithms where the parameters of the process and disturbance models are estimated in a single-step. Most estimation schemes suffer from an inherent weakness of slow convergence of the estimated disturbance model and as a result fixed user specified approximating polynomials are often employed in practice. The proposed two-stage method avoids the problem of slow convergence and does not require any prior knowledge about the disturbances, and is found to work excellently in simulations. The predictive controller is based on a long-range prediction of the output using a Kalman filter predictor, and minimizes a user specified performance index. The cost function used provides a set of simple tuning knobs which can be used to stabilize any given process and which can give desired control performance. An expert system is proposed to direct the overall control algorithm and a simple rule base was proposed and tested using FORTRAN for controlling the two-stage identification.

2) The second most significant contribution of the thesis is the development of an improved least squares (or ILS) recursive parameter estimation algorithm. The standard RLS algorithm which is widely used in process identification and adaptive control suffers from a number of problems such as d.c. bias, turnoff, covariance blowup, parameter drifting, lack of excitation and numerical problems. In ILS these problems were solved using the following steps: data

preprocessing such as mean elimination, low-pass filtering, normalization and scaling; constant trace via a variable forgetting factor; on/off criteria to stop estimation when the excitation is poor and a parameter projection facility to constrain the parameters to known regions. A notable feature of the ILS is that most of the proposed steps are general in scope and are required in some form or the other in any reliable parameter estimation scheme. The ILS algorithm is used in the model identification step of the proposed adaptive control scheme.

3) The third significant contribution of the thesis is the development of MOCCA, a non-functional step response model based multi-step predictive control algorithm that can be applied to non-square, multivariable industrial processes in the presence of constraints. It is based on the same general principles as DMC, MPHC and MAC; but the major emphasis of the work is on a more general formulation; on developing computationally efficient techniques to solve the various optimization problems; and on the use of feedback prediction for improved disturbance rejection. Since the step response data can be easily generated the algorithm can be brought on-line with a minimum of effort. Also step response data obtained directly from the plant can include mild non-linearities of the process and gives improved or more robust performance compared with schemes based on minimal (or parametric) linear models. Note that the supervisory system in the proposed adaptive controller generates the reference signal

which ensures that system constraints are satisfied by solving an on-line optimization problem. Since MOCCA employs a constrained optimization procedure its software can be directly used in the supervisory subsystem of MAPC.

4) The inverse interactor matrix characterizes the "delay structure" in discrete MIMO systems and hence allows the control techniques developed for discrete SISO systems with time-delays to be generalized to the multivariable case. In particular the role of the inverse interactor matrix in designing feedback (e.g. Smith predictor type control schemes) and feedforward controllers for MIMO systems has been studied. A knowledge of the inverse interactor is required for designing one-step and/or multi-step predictive controllers and also contains the prior knowledge needed in designing MIMO adaptive controllers. The inverse interactor plays a natural role in a multivariable extension of proposed MAPC.

5) AI and knowledge based systems are currently hot areas. They have great potential for applications in the process industry in general and for developing control systems in particular. The role of AI and KBS was demonstrated by using AI and fuzzy logic to obtain time-optimal servo control and regulatory control in the presence of constraints of a simple process unit. The AI controllers were implemented on an IBM PC, in the AI programming language PROLOG. The experimental results obtained using these controllers are included. Emphasis was given to RTSB applications of AI. A list

of characteristics that are important in RTSB AI applications was presented. Expert systems provide a convenient and appropriate tool to implement the executive system in MAPC to direct its overall operation.

As a final note it must be pointed out that MAPC was developed incrementally starting with MOCCA, investigation of the role of the inverse interactor, study of the use of AI and development of ILS; and MAPC integrates these individual pieces of the work into a single adaptive process control system.

7.2 Recommendations for future work

1) The adaptive control algorithm, MAPC, proposed in this thesis was presented only for the SISO case. A few preliminary simulations were presented which show the potential of MAPC in solving the major problems associated with the application of adaptive control to industrial processes. More simulations and even experimental evaluations are, however, required to fully understand its utility. A detailed comparitive study with a related class of algorithms such as GPC may be fruitful. The preliminary evaluation also justifies its extension to the multivariable case.

2) The major issues that may crop up in extending MAPC to the multivariable case are the handling of time-delays, problem of multivariable interactions and their decoupling, and of course the identification of the

multivariable process. Each of these problems is a separate topic of research by itself. The multivariable case requires a knowledge of the process "delay structure" and the results presented in Chapter 4 on the interactor factorization may be used here as a starting point. Since MAPC is based on a long-range prediction of the process outputs, its extension to the MIMO case does not require a full knowledge of the inverse interactor matrix. One needs to know only the delay and the steady state gain of each of the (1,1) minors in the process transfer function matrix. If the inverse interactor is diagonal, which fact can easily be deduced from this information, it is straightforward to extend MAPC to the MIMO case. Otherwise, as discussed in Chapter 4 a pre-compensator can be added to the process so that the compensated process model has a diagonal inverse interactor. Note that the pre-compensator may introduce extra delays into the closed-loop. The identification of the process can be done by decomposing a pxm MIMO process into p-MISO systems. For each of these p-MISO systems, MVMAPC requires a process model and a residual model. A diagonal inverse interactor automatically leads to complete decoupling when the controller is dead-beat. However, complete decoupling is often not needed or may even be undesirable (cf. requires severe control action). The trade-offs involved depend on the problem; and the controller parameters in MAPC can be used to achieve a desired degree of decoupling. Note that MVMAPC formulation and its complete evaluation is a major piece of work.

3) No theoretical stability results are available for MAPC at the present stage and obtaining some concrete results can be another important area for future work. Linking MAPC with the pole placement approach may be fruitful and can help in obtaining the needed theoretical results for MAPC.

4) An AI executive for MAPC was outlined and only a part of the system (i.e. to control the identification step) was implemented and tested. Developing a complete AI executive system along the lines presented in Chapter 6 is another area for future work. A realistic development of this system requires a sophisticated AI work station and a good knowledge of AI tools and techniques. The HP/9000, LISP based AI work station in the Department is an excellent facility for this work.

5) The supervisory system for MAPC was presented but not evaluated, especially in the presence of constraints. The advantage of the supervisory system may not be appreciated in the SISO case but in the MIMO case it is expected to play a significant role.

6) The complete software development of MAPC (i.e. all the sub-systems presented in Figure 6.1) and its evaluation is again another significant area for future work. It is conceivable to develop a commercial scale system with the computer and experimental facilities available in the Department.

7) The stability and robustness properties of MOCCA type control algorithms is another key area for future work. Combining a Kalman filter with MOCCA (i.e. for feedback

prediction to reject disturbances, MPM etc.) can be very useful.

8) The ILS algorithm presented in this thesis is a significant improvement over standard RLS for parameter estimation, especially in the self-tuning context. It, however, opens up a whole area of further study as to what is the best parameter estimation scheme in process identification. It is also important to investigate other related areas such as: use of models with a balanced realization in process identification (vis a vis scaling) , directional forgetting, and various interpretations of the covariance matrix (i.e. the covariance ellipsoids) in RLS.

8. Nomenclature

8.1 Technical abbreviations

MOCCA	Multivariable, Optimal, Constrained Control Algorithm
ILS	Improved Least Squares
DMC	Dynamic Matrix Control
MAPC	Multi-step Adaptive Predictive Controller
MVMAPC	Multivariable Multi-step Adaptive Predictive Controller
GPC	Generalized Predictive Controller
SISO	Single-Input Single-Output
MIMO	Multi-Input Multi-Output
AI	Artificial Intelligence
ES	Expert System
KBS	Knowledge-Based System
MPHC	Model Predictive Heuristic Control
IDCOM	Identification and Command
MAC	Model Algorithmic Control
SP	Smith Predictor
LQG	Linear Quadratic Gaussian
GMV	Generalized Minimum Variance
STR	Self-Tuning Regulator
STC	Self-Tuning Control
GPP	Generalized Pole Placement
RLS	Recursive Least Squares
RELS	Recursive Extended Least Squares
TCS	Turnbull Control Systems
DDC	Direct Digital Control

TT	Temperature Transmitter
PC	Personal Computer
NMP	Non-Minimum Phase
LS	Least Squares
ELS	Extended Least Squares
MPM	Model-Process Mismatch
GLS	Generalized Least Squares
CUSUM	Cumulative Sum
ROM	Reduced Order Modeling
APCS	Adaptive Predictive Control System
IF	Interactor Factorization
OR	Ogunnaike and Ray
GMDC	Generalized Multi-Delay Compensator
IMC	Internal Model Control
QP	Quadratic Program
LP	Linear Program
CARMA	Controlled Auto-Regressive Moving Average
CARIMA	Controlled Auto-Regressive Integrated Moving Average
ARMA	Auto-Regressive Moving Average
AR	Auto-Regressive

8.2 Nomenclature for Chapter Two

Alphabetic

y	process output
u	control input
u_0	initial steady state input in the optimal control policy
u_{max}	maximum control input in the optimal control policy

u_{min}	minimum control input in the optimal control policy
u_{ss}	steady state control input in the optimal control policy
y_{sp}	setpoint
e	tracking error $(y_{sp}-y)$
e^*,N	switching parameters in the optimal control policy
dy/dt	rate of change of y
k_p	process gain
y	derivative of y
y_c	constraint on y
k_c	feedback controller gain
T_i	integral gain in the feedback controller

Greek

$\mu_x\{y\}$	membership value of the fuzzy set "y is x"
Δ	differencing operator
Δx	increment of x

8.3 Nomenclature for Chapter Three

Alphabetic

$\{x(i), i\epsilon[1,N]\}$	vector $(x(1), \ldots, x(N))^T$; shortened to $\{x\}$
$\{y_m(t+i/j), j\epsilon[1,P]\}$	model based prediction of y based on inputs upto and including j

$\{y_d(t+i/j)\}$	reference trajectory specified at time j
y	measured process output
u	absolute value of process input
y_{sp}	setpoint
y_m	model output
y_d	desired output
$\tilde{y}$	modeling error $(y-y_m)$
y_m^*	model based prediction of output using measured inputs
e	tracking error (y_d-y)
$\{a_i, i\epsilon[1,N]\}$	process unit step response
a_{ss}	steady state value of unit process step response
N	length of the step response in number of sampling intervals
A_1, A_2	dynamic matrices
A^*	controller matrix
P	output horizon
M	control horizon
k	process pure dead time
m_u	number of process inputs
y_{md}	effect of measured disturbance at the output
v	effect of unmeasured disturbance at the output
d	disturbance input
C,A,F,G	polynomials in q^{-1}
c_2	evaporator outlet concentration
w_1	first effect level
w_2	second effect level
B_1	first effect bottoms

B_2	second effect bottoms
F	feed flow rate
S	steam flow rate
G	arbitrary matrix
Z	arbitrary vector

Greek

Γ_y, Γ_u	weighting matrices
Δ	differencing operator
β	reference trajectory filter parameters
α	feedback filter parameters
ω_i^m	feedback parameters for prediction at t+m
$\gamma_{y_i}, \gamma_{u_i}$	weighting constants

Superscripts

T	transpose

Subscripts

sp	setpoint
m	model
F	filtered
d	desired
ss	steady state
md	measured disturbance
y	output or tracking error
u	input

8.4 Nomenclature for Chapter Four

Alphabetic

y	scalar process output
u	scalar process input
y_{ref}	scalar reference input
y_P	scalar predictor output (cf. Figure 4.2)
v	scalar disturbance input
d	process dead time ($=k+1$)
k	process pure dead-time
q^{-1}	backward shift operator
s	laplace operator
$A(q^{-1}),B(q^{-1})$	scalar polynomials in q^{-1}
n,m	degrees of polynomials
y	multivariable process output
u	multivariable process input
y_{ref}	multivariable reference input
y_P	multivariable predictor output
v	multivariable disturbance input
$\hat{y}$	filtered y
$\hat{y}_{ref}$	filtered y_{ref}
$T(q^{-1})$	transfer function matrix in q^{-1}
$T'(q)$	transfer function matrix in q
$R_T(q^{-1})$	residual matrix in IF
$\hat{R}_T(q^{-1})$	residual matrix with $\hat{R}_T(1)=T(1)$
K_T	high-frequency gain matrix of $T(q^{-1})$
$H_T(q^{-1})$	a unimodular matrix
T_m	process model
P	predictor (cf. Figure 4.2)
T_C	controller

T^*	T with delays of all (1,1) minors removed
l	number of process inputs
p	number of process outputs
$T_F(q^{-1})$	factored transfer function matrix in GMDC
T_+, T_-	factors of T in IMC design
$T_C^{FF}(q^{-1})$	feedforward controller
$T_L(q^{-1})$	load transfer function matrix
$D(q^{-1})$	pre-compensator used in GMDC
T_{ref}, T_{ref_1}	reference model transfer function matrices
$A(q^{-1}), B(q^{-1})$	polynomial matrices in q^{-1}

Greek

$\theta, \theta_1, \theta_m$	scalar transfer functions
$\alpha(q^{-1}), \beta(q^{-1})$	polynomial matrices in q^{-1}
$\xi'_{T'}$	interactor of T'
ξ_T	inverse interactor of T

Superscripts

FF	feedforward

Subscripts

ref	reference
m	model
L	load
C	controller
F	factored

8.5 Nomenclature for Chapter Five

Alphabetic

$P(t)$	covariance matrix
Y	measured process output
U	absolute value of control input
e	perturbation input
d,d'	d.c. bias
G_0	plant transfer function
G_1	perturbation transfer function
A,B,C	polynomials in q^{-1}
$Y_{d.c.}$	d.c. value of Y
$U_{d.c.}$	d.c. value of U
y	mean-deviation of filtered output
u	mean-deviation of filtered input
n,m	orders of A and B
K	estimator gain
$V(\hat{\theta})$	estimator loss function
S	scaling matrix
U_f	filtered U
Y_f	filtered Y
$n(t)$	normalization factor
P,Q	scaling matrices
x,x',b	arbitrary vectors
A,A'	arbitrary matrices
$C\{A\}$	condition number of A
Y	measurement vector in batch least squares
y^n	normalized y
$Q(t)$	square-root of P(t)
$P_{s(t)}(t)$	P(t) scaled by S(t)
tr P	trace of P

r	a scalar constant
$\dim(x)$	dimension of x
$\lVert x \rVert$	euclidean norm of x
I	identity matrix
w_i	i^{th}-eigenvector
$\lVert x \rVert_2$	2-norm or euclidean norm of x
$\lVert x \rVert_\infty$	∞-norm of x
$\lVert P \rVert_2$	induced 2-norm of matrix P
$\lVert P \rVert_\infty$	induced ∞-norm of matrix P
$V(t)$	Lyapunov function
$e(t)$	*a priori* equation error
R	radius of the parameter projection region
$F(t)$	increasing sub-sigma algebras
P_{max}, P_{min}	scalar constants
M	scalar constant
A	scalar constant and also a matrix or a polynomial in q^{-1}

Greek

λ	forgetting factor
τ	process pure dead time
$\hat{\theta}$	estimated parameter vector
ϕ	regressor vector
ξ', ξ	modeling error
η	perturbation term
θ_*	true parameter vector
θ	parameter error vector $(\theta_* - \hat{\theta})$
λ_u	forgetting factor in the estimation of $U_{d.c.}$
λ_y	forgetting factor in the estimation of $Y_{d.c.}$

θ	parameter vector estimated in batch least squares
ϕ_n	normalized regressor vector
$\phi_{nS(t)}$	normalized and scaled regressor vector
Φ	data matrix in batch least squares
α_1, α_2	scalar constants
$\mu_i\{A\}$	i^{th}-eigen value of matrix A
$\iota, \kappa, \delta, \epsilon, \gamma$	scalar constants
$\hat{\theta}_p$	projected parameter vector
θ_c	center of parameter projection region

Superscripts

T	transpose
n	normalized

Subscripts

d.c.	direct current or mean
f	filtered
S	scaled
n	normalized
nS	normalized and scaled

8.6 Nomenclature for Chapter Six

Alphabetic

y	absolute, measured value of process output
u	absolute value of process input
k	process pure dead time

r	an auxiliary measured input (for feedforward elimination)
χ	disturbance or residual signal
e	perturbation input
$\bar{u}$	mean value of u
$\bar{y}$	mean value of y
d',d	d.c. bias or load-disturbance
$A,B,C,A',B',C',$ D,E,F,G,L,N,T	polynomials in q^{-1}
$n_a',n_b',n_c,n_g,$ n',n_f,n_d	orders of various polynomials
$\mathbf{x}$	process state vector
ξ	disturbance state vector
$\mathbf{z}$	complete state vector
H_1,H_2,H	matrices in the state-space model
y_m	process model based prediction of output
R_w,R_v	noise covariances
K	Kalman gain
$\bar{K}$	steady state Kalman gain
P	state prediction error covariance
$\bar{P}$	steady state state prediction error covariance
S	scaling matrix
$\tilde{y}$	mean-deviation of filtered y
$\tilde{u}$	mean-deviation of filtered u
$\hat{\mathbf{x}}$	estimate of $\mathbf{x}$
$\hat{\mathbf{z}}$	estimate of $\mathbf{z}$
$\hat{y}$	total prediction of process output
y_m^*	open-loop process model based prediction of y using measured values

y^*	closed-loop prediction of y using measured values (i.e. $y_m^*+\hat{d}$)
$\hat{d}$	estimate of d
$\{x(i), i \in [1,N]\}$	vector $(x(1), \ldots, x(N))^T$; shortened to $\{x\}$
A, A^*, S	constant matrices
N_1, N_2, N_y, N_u	constant integers
J	controller performance index
y_{ref}	reference input
y_{sp}	setpoint
g	constant vector
$\hat{y}_{ref}$	modified reference input
u^-, u^+	limits on control signal
h^-, h^+	constant vectors
S_i, T_i	CUSUMs

Greek

$\Phi_1, \Phi_2, \Lambda_1, \Lambda_2,$ $\Lambda', \Gamma_1, \Gamma_2, \Phi, \Lambda, \Gamma$	matrices in the state-space model
λ	scalar constant
ξ	disturbance state vector
$\hat{\xi}$	estimate of ξ
Γ_y, Γ_u	diagonal weighting matrices
η	perturbation term
$\hat{\theta}$	estimated process model parameters
$\hat{\theta}_r$	estimated residual model parameters
ϕ	regressor for process model estimation
ϕ_r	regressor for residual model estimation
$\gamma_{y_i}, \gamma_{u_i}$	weighting consants
Δ	differencing operator

Superscripts

T	transpose

Subscripts

ref	reference
sp	setpoint
y	pertaining to y
u	pertaining to u
w	pertaining to w
v	pertaining to v
r	residual